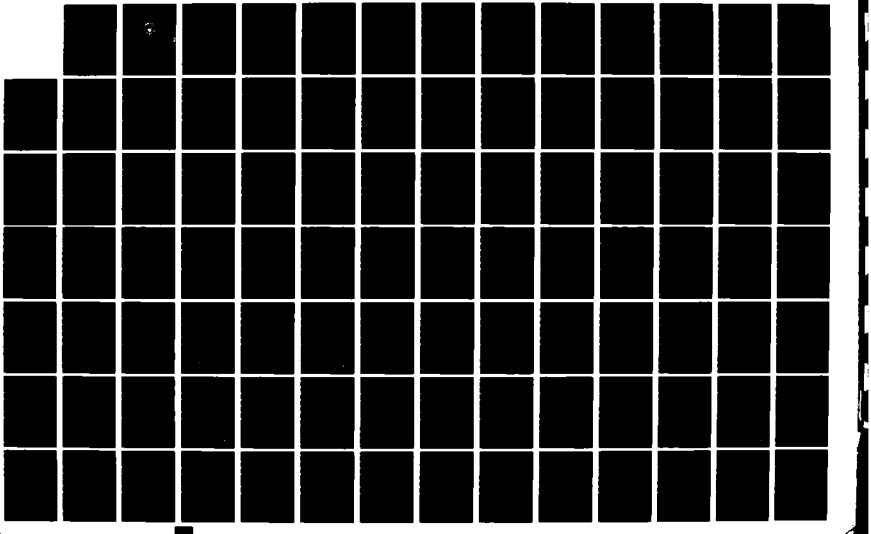
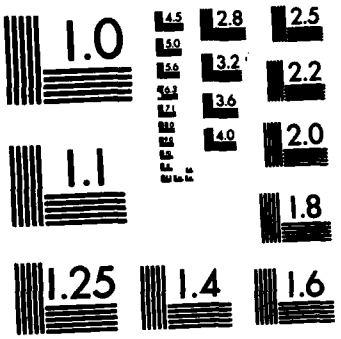
AD-A124 656 TEMPOA: AN INTERACTIVE SIMULATION FOR THE APPLE III MICROCOMPUTER(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA C D OWENS OCT 82 1/2

UNCLASSIFIED F/G 9/2 NL





# NAVAL POSTGRADUATE SCHOOL
## Monterey, California






# THESIS

TEMPOA: AN INTERACTIVE SIMULATION
FOR THE
APPLE III MICROCOMPUTER

by

Christopher D. Owens

October 1982

Thesis Advisor: Alvin F. Andrus

Approved for public release; distribution unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. AD-A124656 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) TEMPOA: An Interactive Simulation for the Apple III Microcomputer | | 5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; October 1982 |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Christopher D. Owens | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940 | | 12. REPORT DATE October 1982 |
| | | 13. NUMBER OF PAGES 127 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | | 15. SECURITY CLASS. (of this report) Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | | |
|---|---|---|
| TEMPO | Computer | Gaming |
| TEMPOA | Program | |
| Apple | Simulation | |
| Microprocessor | Wargame | |
| Microcomputer | Game | |

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

TEMPOA is a computer interactive version of the original General Electric TEMPO game, and is written for the Apple III microcomputer. A minimum of two players vie for a strategic 'win' through judicious budgetary planning in the development and procurement of realistic weapon systems. All decisions are subject to the constraints of limited budgets, hardware

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

inventories, research and development time lags and the uncertainties of war and inflation. A third party, assuming the role of game umpire, selects all simulation parameters and oversees the play of the game.

Approved for public release; distribution unlimited.

# TEMPOA: An Interactive Simulation for the Apple III Microcomputer

by

Christopher D. Owens
Lieutenant, United States Navy
B.S., Marquette University, 1975

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN OPERATIONS RESEARCH**

from the

NAVAL POSTGRADUATE SCHOOL
October 1982

Author _____

Approved by: _____
Thesis Advisor

_____
Second Reader

_____
Chairman, Department of Operations Research

_____
Dean of Information and Policy Sciences

## ABSTRACT

TEMPOA is a computer interactive version of the orignial General Electric TEMPO game, and is written for the Apple III microcomputer. A minimum of two players vie for a strategic win through judicious budgetary planning in the development and procurement of realistic weapon systems. All decisions are subject to the constraints of limited budgets, hardware inventories, research and development time lags and the uncertainties of war and inflation. A third party, assuming the role of game umpire, selects all simulation parameters and oversees the play of the game.

TABLE OF CONTENTS

LIST OF FIGURES

# I. INTRODUCTION

The simulation exercise TEMPO is an example of the historical interest shown towards wargaming by the U.S. military. The TEMPO military planning game was conceived and designed by the Technical Military Planning Organization (TEMPO) of General Electric Company of Santa Barbara, California, in the early sixties as a strategic budget planning wargame.

While originally a manually played simulation, TEMPO was designed to pit the budgetary planning and decision making skills of opposing teams against one another, victory being achieved when one team had gained a clear superiority of capabilities at game end. Both the measures of superiority and the goal of the exercise are two aspects of the game which have undergone changes during the ensuing two decades since its original inception.

Receiving wide dissemination in both civilian and military communities, the game has been used by the Defense Resources Management Course at the Naval Postgraduate School; the U.S. Army Management School at Ft. Belvoir, Virginia; the Air War College, Air Command and Staff College, and Squadron Officers' School all at Maxwell Air Force Base, Alabama. As recently as 1981 TEMPO was introduced as a software simulation on the PDP 11/70 computer in the Naval Postgraduate School's Command, Control and Communications (C3) Laboratory.

TEMPOA, the subject of this thesis, is an adaptation of a version of TEMPO to the Apple III microcomputer. Considered by many to be 'state-of-the-art' at this date, the Apple III with its 128K expandable memory lends itself to large scale programming in a way never before available to such a vast audience.

As a software device for the Apple III, the TEMPOA program will enable users to simulate the decision making process and resource management dilemmas embodied in the weapons development and procurement processes. While an attempt has been made to accurately replicate annual budget overheads, research and development constraints, time limitations and the uncertainties of war and inflation, the game is only a facsimile and falls short of realism on several counts, as detailed later. However, the highlights of the original game and the major influences in the budgeting process are contained in TEMPOA, and the adaptation is an enhancement in many ways over previous versions.

## II. GENERAL DESCRIPTION

TEMPOA requires at a minimum two players and an umpire. The players align themselves into two teams and alternate during the game in making annual procurement and investment decisions. The game simulates a period of time of up to twenty years, and can be terminated at the close of any year.

Player decisions revolve about the development and procurement of new weapon systems (research and development), the buildup or scrapping of systems existing in current inventories and the investment of funds into intelligence gathering activities. Decisions are transformed into expenditures, which in turn promote a player's global capabilities. These capabilities are directly assessed in terms of weapon system 'utils', a single measure assigned each unit of a particular system.

Each player may be faced with choices affecting up to thirty weapon systems during the play of the game. Systems are of two basic sorts, offensive or defensive, and are further subdivided into two categories, weapon system type A (strategic) or weapon system type B (tactical). A type B defensive weapon system may only be used to defend against a type B offensive weapon system. Type A systems are similarly constrained.

At the conclusion of each year, after both players have completed and entered all decisions for the year, TEMPOA computes an annual winner based on gross offensive weapon system utility points accumulated by each player for systems in inventory, discounted by the appropriate defensive system utility values amassed by his opponent. If a large discrepancy exists between player standings, then there exists a significant possibility of war, as computed by an algorithm detailed later. A war translates to a healthy budget cut for the player with the lesser net utility score.

The role of the umpire is twofold; to initialize all game parameters at the outset, and to monitor the annual play. A major enhancement in the current version of TEMPOA is the degree of control exercised by the game umpire in selecting game parameters. The term 'game parameters' loosely refers to individual system parameters (twenty total) of up to thirty game weapon systems, the expected values for annual players' operating budgets, and the means by which the war event is to be determined. In all cases, default values are available.

All computations, which in most previous versions were accomplished with tedium and exasperation by hand, are carried out internally by TEMPOA. Pertinent results are displayed immediately to players during the game, and each player is presented annually with the opportunity for obtaining a printed summary of his current inventory and his

budget status (assuming a compatible printer is configured to the Apple III RS232 port or that an Apple Silentype printer is utilized).

At the conclusion of any game year the umpire may select to halt the present game for resumption at a later time. Upon resumption, TEMPOA begins with the game year subsequent to the termination of the original play, and proceeds using all game parameters effective during the last active play. In this manner, play may be periodically postponed to allow for in depth examination of the game situation, a more extensive analysis and forecast of decision consequences, or simply to enable the simulation of several decades of systems development in the game while allowing the players to proceed in their decisions at a more leisurely pace.

## III. THE PLAY

The fcllowing sections detail step-by-step each facet of the game play. The software itself has been designed so as to minimize the amount of preparatory work necessary for the actual play, and no prior information on the part of the players, beyond knowledge of the gaming objective itself, is presumed by TEMPOA. While TEMPOA was written to be self-explanatory and user friendly, a familiarity with this document will enhance a player's performance.

### A. OBJECTIVE

| System Points | Points allotted |
|---|---|
| Under 2000 | Face Value |
| 2000-2999 | 2000 + 90% of amount over 2000 |
| 3000-3999 | 2900 + 80% of amount over 3000 |
| 4000-4999 | 3700 + 70% of amount over 4000 |
| 5000-5999 | 4400 + 60% of amount over 5000 |
| 6000-6999 | 5000 + 50% of amount over 6000 |
| 7000-7999 | 5500 + 40% of amount over 7000 |
| 8000-8999 | 5900 + 30% of amount over 8000 |
| 9000-9999 | 6200 + 20% of amount over 9000 |
| 10000-10999 | 6400 + 10% of amount over 10000 |
| Over 11000 | 6500 (maximum) |

Figure 3.1 Utility Point Discount Rate

Each player begins TEMPOA with the same collection of weapon systems. Each system can be one of four distinct types; offensive type A (OA), defensive type A (DA), offensive type B (OB), or defensive type B (DB). In addition,

when the program initializes, a unique utility value is assigned to units of each weapon system. The gross util value of that system is determined as the product of the system unit util value and the number of units of that weapon system in current inventory. This product is then discounted to reflect the diminishing marginal returns associated with increasing quantities of a given system. The discount rate is depicted in Figure 3.1. Summing the final gross figure over all systems of that type (OA, for example) results in the total player capability figure for that weapon system type.

If system type A is translated to mean a 'strategic' weapon system, and type B a 'tactical' system, then it follows that a type DA system would only be used to counter the opponent's type OA weapon system threat. To carry this notion out mathematically and arrive at a single net utility score for a player, his total util figures for his offensive system types are decremented by his opponent's corresponding defensive util figures. When the defense for a system type exceeds the value of the opponent's offense for that weapon system collection, then the net offensive value of the system type is zero. That is, no credit is given for exceeding the utility figure necessary to neutralize an opponent's offensive capability. Figure 3.2 is an example of how TEMPOA's algorithm is carried out in practice.

Annually, TEMPOA determines which player achieves the greatest net utility figure and declares him winner. Ties are called when opposing net utility figures are equal.

The effects of a player's decisions spill over from one year to the next, and the inventories of a given year may be viewed as histories, cumulative figures which encapsule all previous game decisions. As a result, the annual winner has done more than acquire a victory during that cycle of the game play, and has achieved a cumulative superiority over his opponent. The winner declared at year end, then, is the collective winner of the game to date as well.

The players' goal is to achieve offensive superiority while thwarting his opponent's efforts through an adequate defense. This goal will be frustrated by budget ceilings and the uncertainties as to where the opponent is channeling his resources.

## B. STARTING

The following subsections deal with the initial setup of TEMPOA; program activation, preliminary data entry, and the role of the umpire.

### 1. Preliminaries

The TEMPOA program is booted (initialized) by inserting the 5 1/4" floppy 'TEMPOA DISK 1" diskette into the Apple III's internal disk drive and turning the computer on. The boot is successful when the program logo is displayed on the monitor.

Following the displayed instructions, players will be queried as to whether the current game is a resumption of a previous play, or whether it is to be a fresh game. The intent of this option is explained later. In most cases the game will be a new play.

When players arrive at the point depicted in Figure 3.3 they are asked to enter their respective player/team names. For identification purposes, the program requires each player/team to use a distinct title. The program will distinguish between upper and lower case entries.

Additionally, TEMPOA requires of each player a codeword. This codeword is a keyboard entry which will act as identification verification annually, prior to allowing access to that portion of TEMPOA reserved for recording players' decisions and budget information. The present insertion is the only instance when the entered codeword is visible (and once the RETURN key is depressed, the entered codeword vanishes). Throughout the remainder of the game, entered codewords do not show up on the monitor. It is strongly suggested therefore that chosen codewords be brief, be either all upper case letters or all lower case letters, and lend themselves to easy memorization. There is no recovery for forgotten codewords.

Players will be advised to insert TEMPOA DISK 2 into the external drive. After confirming the insertion by pressing RETURN, a delay of 10-15 seconds occurs while player

```
        Player 1                        Player 2

System      Total               System      Total
 Type       Utils                Type       Utils

  OA         800                 OA        1000
  OB        1000                 OB         900
  DA        1200                 DA        1000
  DB         900                 DB         800

PLAYER 1 NET SCORE:

                Player 1 OA                          800
Sys         - Player 2 DA                          -1000
 A          = Net System A                         = -200
            (if negative, set to 0)                (set to 0)

                Player 1 OB                         1000
Sys         - Player 2 DB                          - 800
 B          - Net System B                         =  200
            (if negative, set to 0)

  NET VALUE = SYS A + SYS B = 0 + 200 = 200
```

Figure 3.2 Player 1 Net Utility Computation

```
Enter the name of Player Number 1 (Then press RETURN):
Enter a codeword for Player 1 (Then press RETURN):

Enter the name of Player Number 2 (Then press RETURN):
Enter a codeword for Player 2 (Then press RETURN):

   ENSURE TEMPO DISK #2 IS LOADED INTO DRIVE #2
            <Press RETURN to Continue>
```

Figure 3.3 Player Data Entry Page

information is recorded and pertinent segments of the main program are loaded into computer memory.

2. Umpire

```
*****************************
*** UMPIRE INSTRUCTIONS ***
*****************************

        * Printer Status *

Enter the number corresponding to printer status:
      1...RS232 UNIT attached and configured to
          receive data
      2...APPLE SILENTYPE UNIT attached
      3...Printer either not connected or not of
          option type
```

Figure 3.4 Printer Selection

The umpire segment and the main code file has been successfully retrieved from DISK 2 and loaded into main memory when the display depicted in Figure 3.4 appears on the monitor, unless the play is a continuation of a previous game. In the latter case, the monitor displays a codeword request page for player access for the year following the point the play was previously interrupted. And in any event, if the computer fails to communicate with the diskette loaded in the external drive, the cue 'FILE TEMPOA: TEMPOA.CODE NOT FOUND' will appear at the top of the monitor, and drive activity will halt. The players should re-insert TEMPOA DISK 2 into the external drive and reinitialize the Apple III.

Following the successful code file loading, the umpire is requested to indicate what type of printer is connected to the Apple III. Both a printed forecast and an exhaustive annual summary of each player's game status (detailed in later sections) are made available at the termination at the end of a player's decision cycle. Neither summary nor forecast will be offered unless the umpire has indicated the presence of a compatible printer. Selection of the SILENTYPE option will assume that the printer is connected to the Apple printer port A. Selection of the RS232 option assumes the printer is properly configured to communicate with the Apple.

Having indicated the printer status, the umpire is presented with the display depicted in Figure 3.5. This menu allows the umpire access to all weapon systems parameters, budget forecasts, and a section designed to allow him to override TEMPOA's algorithmic expectation of war. In

```
* Systems Review *

1...Browse/Alter default weapon system budget
    parameters
2...Add new systems
3...Delete a system
4...Browse/Alter annual budgets for all game years
5...Elect to control the annual probability of war
6...Obtain printout of all game systems and parameters
7...Quit umpire section and begin game play
```

Figure 3.5 Umpire Menu

addition, the umpire may elect to have TEMPOA produce an initial systems printout, if a printer is attached. Such a printout, detailing each parameter setting of all game weapon systems, provides the umpire with complete reference information unavailable to the individual players. The players are only permitted to peruse those systems and R & D candidates which either are available in the current year, or were available to them in some previous year. Opportunity systems which are not yet available are masked from the players by TEMPOA.

A final umpire option is to forgo any changes and quit the umpire game segment, beginning the play with default game parameters. The default settings are discussed individually in the following sections.

Once the umpire quits the umpire game segment, all budget and weapon system parameters are fed identically to both players, so that both begin the play with equal opportunities.

a. Weapon Systems

The umpire may browse or alter the weapon system parameters through appropriate selections from the umpire's main menu (Figure 3.5). From the same menu he may opt to delete entire systems or create new systems. A total of thirteen totally new weapon systems may be manufactured by the umpire, and can augment the seventeen default systems resident in TEMPOA. Since any or all of the parameters in the default

SYSTEM 2

A Name.....OA2

B Type.....Offensive System Weapon Type A
C First year R&D can start:.....1
D Yrs R&D completed at game start:.....0
E First R&D year cost:.....$400
F Second R&D year cost:.....$900
G Third R&D year cost:.....$700
H Earliest year available (after R&D):.....4
I Units in inventory (at game start):.....0
J Acquisition cost (per unit):.....$300
K Operating cost (per unit):.....$175
L Value in utils (per unit):.....30
M Maximum annual purchase rate:.....15

Enter one of the FOLLOWING options:

A...Delete displayed system and advance page
B...Advance page without deleting displayed system
C...Quit deleting

Figure 3.6 Umpire Systems Browse

systems may be accessed and altered by the umpire, the total number of configurable weapon systems for game play is thirty.

Figure 3.6 shows the display that the umpire is presented with when the Browse/Alter option is used. An explanation of each system parameter follows.

'System Name' is any alphanumeric string of characters. The seventeen default systems are loaded initially with generic names, such as OA1, DB2, etc., but may be umpire altered to read HARPOON, MX1, etc. to enhance the flavor of the game.

'System Type' refers to which of the four categories the displayed weapon system belongs.

The next five program lines deal with research and development. Normally, all systems which are not in inventory must be acquired through an R & D process. However, the umpire has the option of making available systems not requiring R & D. This option is exercised by making the appropriate entries in Figure 3.6. (If the umpire elects to use the default systems,only four will appear in inventory at game start.)

'First Year R & D Can Start' refers to the game year in which the opportunity for initiating the project is first available. 'Yrs R & D Completed at Game Start' allows the umpire to begin the game with R & D partially completed on selected systems. The next three lines indicate the annual expenses of conducting R & D on a system. An R & D project may last from one to three years. To account for inflation and possible cost overruns, a uniformly distributed random variable is drawn from the discrete set {0, 50, 100, 150, 200}. The 'actual' annual expense of R & D is the sum of this variable and the umpire's elected R & D expense figure. This result will be made known to the players as the game progresses.

The "Earliest Year Available (After R & D)' refers to the first year in which the player may add the system to his inventory. Adding the number of required years of R & D to the first year R & D can start should result in the first year the system becomes available. TEMPOA conducts

this simple check on all systems created or altered, and will alert the umpire of the discrepancy when an attempt is made to either advance the page in the 'Browse/Alter' option, or quit the umpire segment.

'Units in Inventory' is applicable only to those systems in inventory at game start and not requiring R & D.

'Acquisition Cost' is the purchase price per unit, and is assessed only once during the game play upon unit procurement.

'Operating Cost' is the annual cost of maintaining and operating a unit in inventory. This is an annual assessment. Units which are in inventory are assumed to be operated, and are so assessed.

'Value in Utils' is simply the unit util value, and when multiplied by the number of units in inventory, produces the gross util value of the weapon system.

A ceiling exists for the number of units of a specific weapon system which may be procured during any given year. This is referred to as 'Purchase Rate' and is the final system parameter listed in Figure 3.6.

b. Budget

Each player begins his year with the same budget figure as his opponent. This amount is based on the budget figures elected by the umpire. Figure 3.7 depicts the page on which this election is made, and is accessed through the umpire's main menu. The values presented are the default

BUDGET INFORMATION

| YEAR | AMOUNT ($) | YEAR | AMOUNT ($) |
|---|---|---|---|
| 1 | 9300 | 11 | 9300 |
| 2 | 9600 | 12 | 9600 |
| 3 | 9400 | 13 | 9400 |
| 4 | 9200 | 14 | 9200 |
| 5 | 9100 | 15 | 9100 |
| 6 | 9100 | 16 | 9100 |
| 7 | 9100 | 17 | 9100 |
| 8 | 9100 | 18 | 9100 |
| 9 | 9100 | 19 | 9100 |
| 10 | 9100 | 20 | 9100 |

<Enter 'C' to make a change>
<Enter 'Q' to quit>

Figure 3.7 Umpire Budget Review

budget values, and the years shown span the maximum duration of the game, twenty years. (The umpire is queried annually whether or not to conclude the game.)

TEMPOA deviates from the umpire elected budget figure in the following manner: In an attempt to simulate effects of inflation, budget cuts and budget increases, a uniformly distributed random variable is drawn annually from the interval [-1000, 500]. The actual budget figure presented to and utilized by the players is the umpire-elected value (Figure 3.7) summed with the random drawn variable. (Budget surpluses are not carried over into the following year.) This final figure, discounted by penalties accrued by the player, is displayed in several locations to each player during the year in which the figure is applicable. (Penalties are discussed later in the Penalties

section.) In addition, the umpire's elected figure for the following year is displayed to the players in the annual Budgetary/Systems Purchase Status tableau, explained later.

c. Probability of War

At the end of each year, after both players have completed their decision cycles, TEMPOA computes the annual probability of war. First the program determines the player with the larger net utility capability (as described in the Objective section) and declares him game winner for the year. It then algorithmically computes the ratio of the smaller net utility figure over the larger, and subtracts the results from unity. The final figure is the TEMPOA annual probability of war. Note that the larger the disparity between player capability, the greater the resulting probability of war.

PROBABILITY OF WAR COMPUTATION OPTION

The annual possibility of a war is determined probabilistically by a random draw whose threshold value is based on the magnitude of disparity between players' total adjusted utility figures.
The umpire may elect to override the above computation method and be queried annually as to whether the war event is to take place.

OVERRIDE (Y or N)

Figure 3.8 Umpire's Probability of War Option

The umpire may elect, through an option available in his main menu (and depicted in Figure 3.8), to let TEMPOA generate a uniformly distributed random variable (between 0

and 1) to determine whether the war event occurs. Or he may elect, through the same option, to have TEMPOA generate and display the computed probability of war, and query the umpire annually to determine whether the event should take place. This override capability defaults to 'off', and TEMPOA will, unless otherwise directed, generate the war event independently by the algorithm detailed above. Once the war event occurs, the player with the larger net utility figure is declared the winner.

## C. ANNUAL PLANNING

The following sections treat what has been termed the player 'decision cycle', which begins with the player codeword entry. The decision cycle concludes when a player elects the 'Quit' option from his main menu (Figure 3.9).

### 1. Annual Forecast

Following the entry of a player's codeword, he is provided the opportunity of obtaining a printout (if the umpire has previously indicated a printer is attached) detailing the imminent year's weapon systems and budget status. Such a printout would be useful if the player were acting as terminal operator for a team of game participants.

The forecast has eight segments. The contents of each segment are only briefly described below. A more detailed discussion is provided later in this section.

The first segment is a list of all weapon systems currently in inventory. The second is a similar listing of

all systems available for purchase but currently not in inventory. Cumulative utility figures for each of the four main systems types are contained in segment three. New R & D opportunities, continuing R & D opportunities, and shelved R & D opportunities make up the next three segments. Segment seven displays the information gathered as a result of the previous year's intelligence expenditures. The final segment provides the working budget for the current year.

2. Main Menu

All expenditures made by a player are made through the main menu. This menu supplies a common return point following any procurement or investment decision, and offers the sole cycle escape avenue through its 'quit' option.

MAIN MENU

1...STATUS OF FORCES CURRENTLY IN INVENTORY
2...STATUS OF FORCES AVAILABLE BUT NOT IN INVENTORY
3...CURRENT RESEARCH AND DEVELOPMENT CANDIDATES
4...INTELLIGENCE/COUNTERINTELLIGENCE INFORMATION
5...CURRENT BUDGETARY/SYSTEMS PURCHASE STATUS
6...QUIT (MAKE NO FURTHER BUDGET DECISIONS THIS YEAR)

Enter the desired option number:

Figure 3.9 Players Main Game Menu

The first four options listed channel the player to those areas where decisions are made affecting his current budget. The fifth option allows the player to view a summary of his budget status, reflecting the immediate impact of all procurement and expenditure decisions. The final option listed signals the completion of the decision cycle.

### 3. Forces Currently in Inventory

Figure 3.10 depicts the display when the first option in the main menu is selected. Those systems with positive inventories are listed. Additionally, the following parameters are displayed; system name, system type, inventory quantity, unit acquisition and operating costs, utils per unit, annual purchase limit, and the total system utility value and operating costs.

FORCES CURRENTLY IN INVENTORY

| SYSTEM(TYPE) | INVENTORY | AQ COST | OP COST | UTILS | PURCH LIMIT | TOTAL OPCOST | TOTAL UTILS |
|---|---|---|---|---|---|---|---|
| OA1(CA) | 20 | $30 | $100 | 60 | 25 | $2000 | 1200 |
| OA3(OA) | 30 | $15 | $60 | 40 | 40 | $1800 | 1200 |
| OA4(OA) | 20 | $40 | $120 | 80 | 10 | $2400 | 1600 |
| OB2(CB) | 5 | $15 | $30 | 20 | 45 | $150 | 100 |

YOU CURRENTLY HAVE $1900 LEFT TO SPEND FOR THE FISCAL YEAR.
<PRESS P TO PURCHASE, S TO SCRAP UNITS OR RETURN TO EXIT>

Figure 3.10 Systems Currently in Inventory Tableau

The player is given three options (from Figure 3.10): purchase new units; scrap existing units; or quit and return to the main menu. If the player elects to either purchase or

scrap units, he is requested to specify which of the systems from the displayed listing he wishes to expand upon (or trim). Once the player has responded, the listing is replaced by a System Information tableau depicted by the upper portion of Figure 3.11. Following TEMPOA's prompting, the player enters the number of units he wishes to buy or scrap. The display of Figure 3.11 is now completed, and an Updated System Information tableau augments the previous System Information tableau. All acquisition and operating costs, as well as annual budget data, are immediately updated to reflect the player's procurement decision. In order to understand the

SYSTEM INFORMATION:

| SYSTEM(TYPE) | INVENTORY | AQ COST | OP COST | UTILS | PURCH LIMIT | TOTAL OPCOST | TOTAL UTILS |
|---|---|---|---|---|---|---|---|
| OA1(OA) | 20 | $30 | $100 | 60 | 25 | $2000 | 1200 |

UPDATED SYSTEM INFORMATION:

| SYSTEM(TYPE) | INVENTORY | AQ COST | OP COST | UTILS | PURCH LIMIT | TOTAL OPCOST | TOTAL UTILS |
|---|---|---|---|---|---|---|---|
| OA1(OA) | 30 | $30 | $100 | 60 | 25 | $3000 | 1800 |

DO YOU WANT TO MAKE ANOTHER CHANGE?
<ENTER Y OR N>

Figure 3.11 Systems Purchase and Update Tableau

updating impact of a decision to purchase or scrap, it is necessary to take a closer look at how TEMPOA performs the arithmetic.

Associated with each system are two special variables, one called the system inventory and the other called the 'threshold' number. The system inventory always reflects the number of units of that weapon system in inventory at any given moment. The threshold number is simply the number of units in inventory with which the player began the year. The second variable represents those units for which an acquisition expense has already been incurred. While the inventory figure may fluctuate throughout the decision cycle, the threshold value will remain fixed, and will provide TEMPOA the necessary flexibility to allow the player to reverse his decision with impunity throughout the cycle.

When a player makes a purchase, his inventory is immediately increased, and his system operating cost is simply the product of the inventory and the unit operating cost value. When a player scraps units of a system, his inventory and system operating costs are immediately decremented. During a purchase, however, he is assessed an acquisition cost for only those units which are in excess of his threshold figure. And during scrapping, he is 'rebated' an amount equivalent to the acquisition costs only for those scrapped systems which are in excess of the threshold figure.

As an example of the numerical role played by the threshold figure, consider the following. A player's threshold is thirty units and his inventory is forty units (a point in the cycle when the player has already purchased ten units since the cycle began). He decides now to scrap fifteen units. His updated inventory will be twenty-five units. His threshold is still thirty units. His updated operating costs reflect his current inventory of twenty-five units. His budget funds remaining has increased by an amount equal to the acquisition price of ten units (not fifteen).

TEMPOA will provide the alert depicted in Figure 3.12 if the player attempts to spend more than he has available. The alert acts merely as a warning, and will not prevent the determined player from overspending his budget. If the player continues, and amasses a deficit, an amount equal to twice the deficit figure is automatically deducted at the end of the decision cycle from the subsequent year's budget.

During any portion of the decision cycle, an attempt to procure more units than the purchase rate allows will prompt an additional warning. TEMPOA then provides the player with the maximum number available for purchase, and will query the player as to his intentions.

WARNING

You have not enough funds remaining in the year to make the desired acquisition

Budget.....$9400

Amount spent previously for acquisitions.....$0
Amount spent for intelligence.....$100
Amount spent for R and D.....$400
Amount to operate all current inventories.....$7200
Amount remaining.....$1700
Acquisition and operating costs
of system quantity selected.....$1860

Deficit.....$160

If you continue with this purchase and do not scrap any other systems (to lower your operating costs) you will have an annual deficit. Twice this amount will then be deducted from next year's budget.
Do you wish to continue with this acquisition?
<Y or N>

Figure 3.12 Budget Overrun Warning Message

4. Forces Available But Not in Inventory

All display formats and purchase procedures in this section are identical with the previous section. Systems itemized in the manner depicted by Figure 3.10 are those whose inventories are zero. Increasing the inventory of any system will immediately cause the system to be transferred from the 'Not in Inventory' category to the 'Forces Currently in Inventory' category. Subsequent perusal of this system requires a return to the main menu, and selection of the 'Forces Currently in Inventory' option.

In the event the forces listing is greater than four systems in length, TEMPOA provides for a series of pages to

be created, each of which expresses system information in the standard format. The paging option is implemented automatically when the need arises, and each page provides the player with the purchase option, and the advance page and quit options. This paging procedure is also implemented in the 'Forces Currently in Inventory' segment.

5. Research and Development Candidates

RESEARCH AND DEVELOPMENT PROJECT STATUS

A...NEW OPPORTUNITIES
B...CONTINUING OPPORTUNITIES
C...SHELVED OPPORTUNITIES
Q...QUIT

Figure 3.13 Research and Development Menu

All systems about which players make decisions can be categorized into three mutually exclusive classes; those systems in inventory, those systems available but not in inventory, and those systems in some stage of research and development. This latter class of systems may be browsed and manipulated through the R & D in the players' main menu. Once selected, an R & D menu, as depicted in Figure 3.13, appears on the monitor.

There are three possible stages in the R & D process. First is the New Opportunity stage. All new R & D opportunities make their initial appearance under this heading, and may be viewed by selecting the first of the options depicted in Figure 3.13. The selection causes the display to be replaced by the tableau depicted in Figure 3.14.

NEW R & D OPPORTUNITIES

| NAME(TYPE) | YRS R&D COMPLETED | TOT YRS REQ'D | YR 1 COST | YR 2 COST | YR 3 COST | ACQ COST | OP COST | UTILS | PURC RATE |
|---|---|---|---|---|---|---|---|---|---|
| DB5(DB) | 0 | 3 | $400 | $350 | $600 | $40 | $100 | 80 | 25 |
| CA4(CA) | 0 | 2 | $250 | $400 | $200 | $30 | $80 | 60 | 40 |
| DB4(DB) | 0 | 3 | $425 | $420 | $400 | $65 | $75 | 55 | 15 |
| DA3(DA) | 0 | 1 | $900 | $650 | $700 | $100 | $120 | 120 | 20 |

* NOTE: ALL NEW R&D PROJECTS WILL BE *
* SHELVED AT YEAREND UNLESS YOU *
* NOW DESIGNATE *

YOU HAVE $1500 REMAINING IN THIS YEAR'S BUDGET.
ENTER THE NAME OF THE SYSTEM WHOSE R&D YOU WISH TO BEGIN:
<TYPE 'QUIT' TO QUIT>

Figure 3.14 New R&D Opportunities Tableau

The information displayed in Figure 3.14 is similar to that displayed in the purchase tableaus with minor exceptions. 'Yrs R & D Completed' is self-explanatory, and will have a value of zero for all new opportunities. 'Total Yrs Req'd' refers to the number of years of R & D required to make the system available for purchase. This figure, when compared to the 'Yrs Completed' value, gives a progress report on the system's R & D status. The maximum number of required R & D years for any TEMPOA weapon system is three. The next several

entries in the tableau reflect the expected cost of pursuing R & D on the described system.

When the player elects to begin or continue R & D on a specific system (the option depicted at the bottom of Figure 3.14), the incurred expense is equal to the figure indicated under the appropriate year column. This cost is immediately deducted from the annual budget. The remaining year cost columns in the R & D menu are forecasts only, and the figures under those headings may vary as a result of an inflation simulation. At the end of each year, a uniformly distributed random variable is drawn from the discrete set {0, 50, 100, 150, 200}, and is added to the subsequent year's expected R & D costs. As a result, the expense of continuing a system's development may become more expensive than anticipated, and players may decide to shelve the project temporarily.

All new opportunity projects about which no player decisions are made will be put automatically into the shelved opportunity class by TEMPOA at the end of the decision cycle. TEMPOA makes no automatic budget assessments for either new or shelved R & D opportunities.

The Continued Opportunity listing, also displayed through the R & D menu, itemizes all systems funded for R & D. The format is similar to that depicted in Figure 3.14. The 'Yrs R & D Completed' value, however, will be non-zero in those instances where R & D was previously funded.

The current budget figure always reflects the expense of pursuing those R & D projects itemized under 'Continuing Opportunities'. When a new opportunity is funded, the system's status changes, and the budget is immediately assessed the appropriate year's R & D cost. Similarly, when a previously shelved opportunity is continued, its status changes and the assessment is again performed (and a possible penalty, described later, is charged).

From the Continuing Opportunities display, a player may elect to shelve one of the listed systems. Shelving a system will cause an immediate reinstatement of budget funds to occur, the amount equivalent to the cost of R & D pursuit for that system. At cycle end, all systems in the Continuing Opportunities status are assumed to remain in that status for the beginning of the subsequent year.

PENALTY PRICES FOR RESUMING PREVIOUSLY SHELVED R&D PROJECTS

| System (Type) | Penalty |
|---|---|
| OA2(OA) | $300 |
| DB3(DB) | $250 |
| OB4(OB) | $275 |
| OA3(OA) | $300 |

You have $1600 remaining in this year's budget.

Enter the name of any system whose R&D you wish to resume:

<Type 'quit' to Quit>

Figure 3.15 R&D Penalty Tableau

Shelved Opportunities, the last stage to which an R & D project may belong, is the set of all systems whose project development is temporarily held in abeyance. The display format for these systems is similar to that of the other two stages. Once shelved, projects are indefinitely held in this status, carried over from one year to the next automatically by TEMPOA. If the player decides to resume development, he is assessed the appropriate year's R & D cost (previously explained), in addition to being assessed a resumption penalty cost. This cost reflects the expense of reactivating old facilities and the general regrouping of effort needed to pursue previously discontinued projects. Specifically, it is equal to $300, or the cost of the most recent year of R & D completed on the shelved project, whichever is less. A listing of penalties for all shelved systems, as depicted in Figure 3.15, is accessed through the third option in the R & D menu.

A budget overrun warning, similar to the purchase overrun warning described earlier, is utilized in the R & D segment, and cautions the player when excessive R & D investments are attempted.

6. Intelligence/Counterintelligence Information

Figure 3.16 displays the intelligence/counterintelligence menu available to the player through his main game menu. The options available are three; receive previously purchased information, fund the purchase of future information, and cancel the funding allotted earlier in the cycle.

SELECT INTELLIGENCE OPTION:

A....Receive Previously Purchased Intelligence Information
B....Purchase Intelligence/Counterintelligence Information
C....Cancel All Intelligence/Counterintelligence Requests for the Current Year
Q....Quit

Figure 3.16 Intelligence/Counterintelligence Menu

Four distinct types of intelligence information may be procured. The first two concern enemy offensive and defensive force strengths, respectively. The latter two are information regarding enemy offensive and defensive R & D projects.

By selecting the first option in Figure 3.16, the player may review information procured by intelligence funding the previous year. In game year one, as in any year not preceded by an intelligence procurement, no intelligence information can be reported, and an appropriate comment to that effect will be indicated. Figure 3.17 depicts typical results of electing the first option from the Intelligence/ Counterintelligence menu (Figure 3.16).

The first two TEMPOA responses in Figure 3.17 will be one of three possibilities: If no money was previously expended for the intelligence category listed, the cue 'NO INFORMATION PROCURED' will be displayed. If the intelligence

```
CURRENT INTELLIGENCE REPORTS:

    ENEMY OFFENSIVE FORCES INFORMATION:
        NO INFORMATION PROCURED

    ENEMY DEFENSIVE FORCES INFORMATION:
        Total enemy  defensive utils are estimated
        to be between 2736 and 3029

    ENEMY OFFENSIVE R&D INFORMATION:
        CIA unable to breach enemy counterintelligence
        barrier

    ENEMY DEFENSIVE R&D INFORMATION:
        Reports indicate enemy currently funding 3
        defensive R&D projects
```

Figure 3.17 Intelligence Report

category was funded, but the enemy provided for counter-intelligence funding, then TEMPOA will respond with 'CIA unable to breach enemy counterintelligence barrier'. If funding was provided and the enemy failed to provide for counterintelligence, then TEMPOA responds with upper and lower bounding estimates of the enemy's total utility for the appropriate system category. Each bound is arrived at separately through distinct uniform random variable draws from the interval [0, 200]. These two variables are then added and subtracted, respectively, from the actual total enemy utility figure to generate the displayed upper and lower bounds.

The last two TEMPOA responses in Figure 3.17 will have the same first two possibilities as explained previously. If funding was provided and the enemy has

established no counterintelligence, however, TEMPOA will respond with the number of currently funded enemy R & D projects of the appropriate system category.

PURCHASE OPTIONS

A...Current Force Strength of Enemy Offensive Forces
B...Current Force Strength of Enemy Defensive Forces
C...Current Offensive Enemy R&D Projects
D...Current Defensive Enemy R&D Projects
E...Supply Counterintelligence Regarding Own Offensive Forces
F...Supply Counterintelligence Regarding Own Defensive Forces
G...Supply Counterintelligence Regarding Own Offensive R&D
H...Supply Counterintelligence Regarding Own Defensive R&D
Q...Quit

Intelligence Cost per Item (A thru D): $100
Counterintelligence Cost per Item (E thru H): $200

ENTER OPTION LETTER:

Figure 3.18 Intelligence/Counterintelligence Purchase Menu

Option two from the R & D menu (Figure 3.16) presents the display depicted in Figure 3.18, and allows the player to procure intelligence information or to provide for the supply of counterintelligence. The cost of intelligence procurement is $100. The cost of providing counterintelligence is $200. Counterintelligence is used to deny the enemy information whose acquisition he is funding. All procurements are immediately deducted from the annual budget figure.

The final option allows the player to reverse previous intelligence/counterintelligence decisions, and cancel all such funding for the cycle.

7. Budget/Systems Purchase Status

BUDGET INFORMATION

Year: 3 Total annual allowance: $9300

Amount spent in acquisitions.....$850
Amount spent on intelligence.....$600
Amount spent on R&D.....$2000
Operating costs by system (those in inventory):

| System (Type) | Quantity | Total Operating Costs |
|---|---|---|
| OA1(OA) | 40 | $1200 |
| DB1(DB) | 100 | $2000 |
| OB3(OB) | 100 | $1500 |

Expected defense budget for next year: $8700
Total monies remaining: $1150

<Press RETURN to Return to Main Menu>

Figure 3.19 Budget/Systems Information

A current summary, depicted in Figure 3.19, is available to the player at any time during his decision cycle through an option in his main menu. The summary provides a documentation of monies allotted during the current cycle for system acquisitions, intelligence and counterintelligence procurement, and R & D funding. Additionally, the summary breaks down all systems currently in inventory by name and system type, and provides individual operating costs. The summary concludes with the amount of budget funds remaining, and an estimate of the following year's budget allocation.

D. ANNUAL OUTPUT

Immediately following the conclusion of a player's decision cycle he is given the option (if the umpire has previously

indicated a printer has been attached) of obtaining an exhaustive printed summary detailing his budget decisions and weapon systems status for the year just completed. Once elected, the printing takes two to three minutes, and cannot be interrupted.

The printout is divided into eight segments. The first segment itemizes all weapon systems currently in inventory, and provides a breakdown of each system into its component parameters. The second segment provides the same breakdown for all weapon systems available but not in inventory. Total cumulative utility figures for each of the four main system types are provided by segment three. The next segment itemizes all continuing R & D projects, providing the same parameter details as contained in the first segment. Segment five treats all shelved R & D projects. As mentioned earlier, this category includes any unacted upon R & D opportunities which were introduced to the player as new during the previous cycle. The sixth segment deals with a budget summary and forecast for the upcoming year. The information provided is similar to that available to the player through his Budget/System Status option in his main menu. The last two segments are forecasts of systems which will appear as opportunities at the start of the following decision year. Segment seven forecasts systems which the umpire has made available in inventory without requiring player R & D effort. Segment eight forecasts new R & D projects to be introduced to the game the following year.

E. WINNING

AS A RESULT OF A TALLY OF
OPERATIONAL FORCES UTILITY
POINTS...

Chris

...IS DECLARED WINNER
FOR GAME YEAR 5.

Figure 3.20 Winner Declaration

TEMPOA declares a win at the end of the year based on the player's net utility figures. The exact algorithm has been explained in the Objective section. Figure 3.20 characterizes the display presented by TEMPOA when a winner is declared. A tie is called in the event the players have identical net utility figures.

F. SAVING THE GAME

At the end of each player's turn, the umpire is presented with a menu selection which provides him the option of terminating the game. Specifically, he may decide to end the game and save the game data, or he may continue with the game. If the umpire elects to conclude the game and preserve the game data, then TEMPOA creates a data file on the external diskette and writes all systems and budget information for each player onto the file. When all disk drive activity has stopped, the play is concluded and the Apple may be turned off. When next the TEMPOA program is initialized, the umpire should indicate

that the game is a resumption (see Preliminaries section for appropriate cue information). In the event the game is again shelved, the old data file on the external diskette is merely updated. The play, then, may be interrupted frequently without loss of data storage space due to a need to save the most current data base. In addition, several separate games may be played in the interval between the saving of a game and its resumption, as long as no attempt is made to save one of the intervening games.

## G. PENALTIES

Three penalties may be assessed a player during the game. The cost of losing a war is $1000, and is immediately deducted from the following year's budget. A second penalty is the extra expense incurred by budget overruns. At the end of each year, twice the amount of any budget overrun is deducted from the guilty player's upcoming year's budget. Note that while a player is penalized for a budget overrun, he is not given credit for budget surpluses. As in many military procurement scenarios, what isn't used is lost. The final way a player may be penalized is by resuming a shelved R & D project. If the shelved system was actively pursued at some earlier time, then an added expense is incurred for project resumption. The cost is equal to $300, or the cost of the most recent year of R & D completed on the shelved project, whichever is less.

It bears mention that, in light of these penalties, a player's actual annual budget may differ drastically from the umpire's elected figure. A budget overrun of $500 coupled with the loss of a war will cut a player's expected figure by $2000. In addition, the random draw made to adjust the budget figure for inflation may slash the budget by another $1000. The resulting budget figure may surprise the unalerted player. Both players are strongly advised to heed TEMPOA-generated budget overrun alerts, and be wary of overextending operating commitments into subsequent years.

## H. STRATEGY

The goal of the game is to gain the upper hand in terms of offensive utility points, while maintaining enough defensive utility points to offset the opponent's offensive capability. Players will not achieve this end simply by funneling monies blindly into offensive capabilities.

If the game duration is predetermined and announced by the umpire, it may be possible to successfully model TEMPOA as a complicated dynamic optimization problem. A variable, undetermined game duration, however, frustrates this approach, and the player is resigned to work out his 'best' strategy with a more primitive scheme.

There are three areas in which a player might sink funds; intelligence, R & D, and system procurement.

The gathering of intelligence information is important for providing guidance for which type of system to allot R & D

funding. And the cost of intelligence procurement is relatively inexpensive when compared to the pursuit of an R & D project or the augmentation of an existing weapon system. Intelligence is as much a priority to a player's enemy as it is to the player himself, hence the need to fund counterintelligence activities. A judicious use of counterintelligence will effectively mask critical information concerning systems where a player is particularly weak, or where a player is expanding in preparation for the great 'coup'.

Most of the default R & D projects require three years of funding before they are available for purchase. A three year delay is a significant interval, and players should not delay in initiating R & D projects which will bear no fruit for several decision cycles. It is easy to become misled by visions of a late return for early efforts. To delay the start of an attractive R & D project in favor of a lesser capable but more immediate investment may be taking risks that the player can ill afford. Frivolous shelving is costly. The penalty price associated with resuming shelved R & D projects (in addition to the delay in getting them 'on the street') may exceed the expense of the budget overrun caused by continuing the project. The costs and penalties for shelving are not always transparent, and the decision to shelve warrants careful scrutiny.

Scrapping systems currently in inventory should be treated in the same way as shelving R & D projects. For each unit scrapped in haste, though the annual operating expense has been reduced, the equivalent acquisition cost has been wasted. Scrapping is an appropriate measure only when presented with an attractive, cost-effective weapon system whose operation under the budget constraint necessitates the replacement of less efficient systems. Systems of the same type may be compared in a cost-effectiveness mini-study, where expenditures are amortized over some predetermined interval. A similar approach has been successfully used to deal with R & D systems.

The brunt of the emphasis in recent TEMPO versions has been on team interaction and the analysis accompanying the decision cycle. In the current version, the duration of the decision cycle is left to the umpire. However, the more time allowed the players during their cycles, the more complex and competitive their analysis is expected to be.

# IV. CONSTRAINTS AND LATITUDES

This chapter is divided into sections. The first section treats player constraints and variable limits, while the second deals with those areas of TEMPOA where random numbers are generated and used.

## A. CONSTRAINTS

Game duration is limited to twenty years. This constraint, elected as a reasonable upper bound for the game duration, was embedded in TEMPOA to allow the umpire the flexibility to alter a set of default budget figures. If a duration greater than twenty years is desired, it is only necessary to alter the length and initialize the values of the integer-valued budget vectors REDBUDGET and BLUBUDGET, declared in the main program.

The maximum number of game weapon systems is thirty. Seventeen are default systems, and may be altered beyond recognition by the umpire. Another thirteen systems may be created by the umpire to augment the default systems. Again, the bound at thirty was elected as a reasonable limit to the number of system alternatives a player would be able to deal with. An incremental increase in the number of systems available increases the game complexity exponentially.

All arithmetic operations performed by TEMPOA use integer-valued variables. And in the main, integer varibles have been used more frequently than long integers. The

probability of war computation, while appearing to use and display real-valued variables, actually uses long integers and an artificial decimal display. As a result of the prevalence of integer variables, the maximum value TEMPOA arithmetic can be expected to deal with is 32767. This is a constraint of the Apple itself. Since the greatest expenditures will not often exceed the annual budget, care need only be taken to ensure the umpire does not elect to use budget values in excess of 32767 (budget default values are approximately 9000). This will artificially bound expenditures. And observing that unit utility values are roughly comparable to unit operating costs, bounding the annual budget figures effectively caps the total systems utility values as well. Annual expenditures and gross utility values are the largest variables dealt with in TEMPOA. Ensuring their bounds are within Apple integer constraints is sufficient to safeguard all other variables.

Aside from the mentioned constraints and the limitations of the display presented on the monitor, TEMPOA has been constructed to be flexible and forgiving. Numerical entries not within the allowable range of the assignment will prompt a TEMPOA cue highlighting the error and cautioning the player.

## B. LATITUDES

Random numbers are drawn and used in four instances during the game play. The Apple System Library unit Applestuff provides the source of the random numbers. All begin uniformly

distributed over the interval [0, 1] . A linear transformation is then performed on each draw to provide the appropriate variable range required. The Randomize procedure, also provided by the Applestuff unit, is used to avoid the repetition of a draw sequence in subsequent games.

Draws are made in the following circumstances; to provide upper and lower bounds to the enemy utility figure reported to the player as the result of an unblocked intelligence request; to determine the annual effect of inflation on the umpire-selected budget value; to determine the effect of inflation on the imminent year's R & D cost for any continuing R & D opportunity; to determine, if umpire-elected, whether a war event is to take place. A more precise description of the algorithms used in the above circumstances is available in the appropriate sections.

## V. RECOMMENDATIONS

TEMPOA is constructed to mime the realities of the defense procurement and development process. To the extent in which it falls short of this goal, room for program enhancement exists.

A glaring artificiality is the encompassing in a single value, called 'util', the full capability and military significance of an entire weapon system. Certainly this streamlines TEMPOA. And it frees the player to concentrate his efforts on budget allocation rather than on pure systems analysis concerns. Nontheless, injecting a more realistic systems assessment would enhance TEMPOA and add a new level of sophistication to the play.

An area not directly addressed in TEMPOA is the cumbersome artificiality of the decision cycle scheme. Forcing one player to 'wait his turn' while the other proceeds through the decision cycle is not merely unrealistic. Boredom is fostered in the nonparticipating player. A more desirable scheme would be a time-sharing arrangement, where both players occupy separate terminals, linked by modem, and conduct their decision cycles simultaneously. Such an arrangement, while posing no great technical difficulties, requires the availability of multiple compatible terminals. In addition, the need would arise for both players to complete their cycles at about the same time, to enable a timely transfer of player status information from

one terminal to the other (in order to provide TEMPOA the necessary data to decide and declare a winner).

An alternate method for dealing with the player cycle is to have duplicate game diskettes given to the players. Each could play simultaneously on separate non-linked terminals. Following each turn, a data file summarizing a player's decisions would be created on each game diskette. The umpire would then halt the game and switch the players' diskettes. Each player could then retrieve the opponent's data file and determine a score. Diskettes would then be returned to the respective players to begin another game cycle.

While this second method allows for simultaneous game play, it requires the additional manipulation of game diskettes. Also, the umpire would be required to monitor the cycle duration and ensure the simultaneous termination of each player's game year.

All of which highlights the desirability of an internal clock in the Apple III. In the current TEMPOA simulation, an umpire takes the place of a clock, and may establish duration limits for player decision cycles. A clock would relieve the responsibility from the umpire, and would also provide a ready solution to the problem posed by the need for simultaneous cycle completion when using two linked terminals.

The above summarizes the more glaring shortcomings of TEMPOA. Other such areas exist, and will become apparent whenever the player finds himself muttering in a frustrated tone of the artificiality of a TEMPOA-generated situation.

# VI. MECHANICS

This chapter discusses the required hardware upon which to play the TEMPOA program. In addition, the section on software dwells on the program makeup, TEMPOA file construction, and procedure segmentation.

## A. HARDWARE

The program TEMPOA requires the use of one Apple III 128K microcomputer, a 120V/60Hz power source, and one external disk drive. While TEMPOA consists of three separate 5 1/4" floppy diskettes, the code required for running the program is contained only on the first two.

The use of a printer is optional. However, in order to obtain an umpire's complete systems summary, an annual player's decision summary, or an annual status forecast, a properly configured printer is essential. When attached, a compatible printer is linked through the Apple RS232 port. If an Apple Silentype printer is used, it must be attached to Printer port A.

## B. SOFTWARE

TEMPOA, as mentioned above, is contained on three diskettes. TEMPOA DISK 1 and TEMPOA DISK 2 contain the program code files and are used to run the game. TEMPOA DISK 3 contains all program text files, and is available to provide the user with the option of reconfiguring or editing the game.

Specifically, TEMPOA DISK 1, which is inserted in the internal disk drive prior to turning the power on, contains the Apple system software necessary to boot the TEMPOA program. Additionally, it contains a data file in which player names and codewords will reside. Using the Apple System Library unit Chainstuff, the booted code file on TEMPOA DISK 1 is linked to the main program on TEMPOA DISK 2, which should be inserted into the external disk drive. Other than the initial diskette insertions, no further file or diskette manipulations are required of the player during the remainder of the program.

An understanding of the rationale behind the TEMPOA file configuration is necessary if the user anticipates altering or enhancing the program text.

File arrangement was dictated by several Apple constaints. First, individual procedure size limits caused simple but lengthy texts to proliferate into multiple smaller units. Hence the final program contains an inordinate number of nested procedures. A second consideration in program design was text file size. In order to amass the quantity of text embodied by TEMPOA, the program was divided into seven text file 'batches', plus the main driver text file. All files are then compiled sequentially using the Apple III Pascal compiler 'include' option.

A constraint affecting program execution was the limitation on the quantity of code that could be loaded at one time

into the main memory. In order to provide adequate space for pertinent sections of the program, large blocks of text were defined as 'segmented' procedures. These procedures, compiled before non-segmented procedures, are not automatically loaded into the main memory, and now remain resident in the code files until called by TEMPOA.

Sheer size of the TEMPOA code file (160 diskette blocks), suggested the use of a third disk drive in program construction and compilation. The requirement that certain system software files be resident in certain drives made TEMPOA file manipulation during construction cumbersome. While users are invited to modify and enhance the current version of TEMPOA, the use of a third drive is heartily recommended for the tasks of editing and compilation.

# COMPUTER PROGRAM

```
UNIT MYSTUFF;
INTRINSIC CODE 17;
                                INTERFACE
USES APPLESTUFF;

(The following three procedures were created and installed as an intrinsic
 unit into the system library.  Each is called frequently by the driver
 program and by its satellite subroutines.)
PROCEDURE PUTIT (x,y : integer; phrase : string);
PROCEDURE CLEARSCREEN;
PROCEDURE HELP;

                                IMPLEMENTATION
PROCEDURE PUTIT;
  (Places a given string at a given screen location.)
  begin
    GOTOXY(x,y);
    writeln(phrase)
  end;                          (putit)

PROCEDURE CLEARSCREEN;
  (Erases entire screen.)
  var
    control : integer;
  begin
    control := 28;
    unitwrite(1,control,2,,12);
  end;                        (clearscreen)

PROCEDURE HELP;
  (Sounds audio tone and displays cue to operator.)
  var
    d : char;
  begin
    clearscreen;
    sound(65,3,63);
    writeln;writeln;writeln;writeln;
    gotoxy(15,10);
    writeln(' You must choose from among the available options.');
    putit(23,18,'Press RETURN to return to menu');
    read(d)
  end;                              (help)

end.                                (mystuff)
```

```
PROGRAM STARTUP;
USES APPLESTUFF, CHAINSTUFF, MYSTUFF;
  {Boots program, provides basic user handshake and introduction.
   Obtains player names, tickler to ensure disk #2 is in drive.
   Chains to main TEMPO program.}

var
  i: integer; component: array [1..4] of string; datafile: file of string;

PROCEDURE DRAW1;
  {Draw1 and Draw2 provide program logo}
  begin
    writeln('                                        WELCOME TO');writeln;writeln;wr
iteln;
    writeln('              TTTTTTTTTT   EEEEEEE   M       M   PPPPPP      000000
    ');
    writeln('              TTTTTTTTTT   EE        MM     MM   PP   PP   00    00
    ');
    writeln('                  TT       EE        MMM   MMM   PP   PP   00     0
0   ');
    writeln('                  TT       EEEEEEE   MMMMMMMMM   PPPPPP   00     0
0   ');
    writeln('                  TT       EE        MM  M  MM   PP       00     0
0   ');
    writeln('                  TT       EE        MM     MM   PP        00    00
    ');
    writeln('                  TT       EEEEEEE   MM     MM   PP          000000
    ');
    writeln('                                                                    ')
  end;                          {draw1}

PROCEDURE DRAW2;
  var
    d : char;
  begin
    writeln('                                                                    ')
;
    writeln('                                                                    ')
;
    writeln('                The interactive game of international warfare financ
e    ');
    writeln('                                                                    ')
;
    writeln('                                                                    ')
;
    writeln('                                                                    ')
;
    writeln('                                Press RETURN to continue
    ');
    writeln('
;
    read(d)
  end;                          {draw2}

PROCEDURE GETNAMES;
  {Obtain player names}
  begin
    clearscreen;
    gotoxy(10,10);
    write('Enter the name of Player Number 1 (Then press RETURN): ');
    readln(component[1]);
```

```
  gotoxy(10,11);
  write('Enter a codeword for Player 1 (Then press RETURN): ');
  readln(component[3]);
  gotoxy(60,11);write('          ');
  gotoxy(10,13);
  write('Enter the name of Player Number 2 (Then press RETURN): ');
  readln(component[2]);
  gotoxy(10,14);
  write('Enter a codeword for Player 2 (Then press RETURN): ');
  readln(component[4]);
  gotoxy(60,14);write('          ')
end;                              {setnames}

PROCEDURE DISK2;
  {Tickler to ensure disk #2 is in Drive #2}
  var
    f : char;
  begin
    putit(18,18,'ENSURE TEMPO DISK #2 IS LOADED INTO DRIVE #2');
    putit(27,20,'(Press RETURN to Continue)');
    read(f)
  end;                              {disk2}

PROCEDURE SCRIBBLE;
  {Introduction}
  var
    d : char;
  begin
    clearscreen;
    putit(13,5,'   The following interactive simulation was written          ');
    putit(13,6,'by Christopher D. Owens, LT, USN, for the Apple III          ');
    putit(13,7,'microcomputer in partial fulfillment of requirements         ');
    putit(13,8,'for the degrees of Master of Science in Operations           ');
    putit(13,9,'Research and Master of Science in Applied Mathematics.       ');
    putit(13,10,'                                                            ');
    putit(13,11,'   Complete program documentation and user instructions     ');
    putit(13,12,'are contained in the thesis document itself, submitted      ');
    putit(13,13,'in September 1982 at the Naval Postgraduate School in       ');
    putit(13,14,'Monterey, California.  The document and accompanying        ');
    putit(13,15,'diskettes are held by Alvin F. Andrus, Department of        ');
    putit(13,16,'Operations Research, at the Naval Postgraduate School.      ');
    putit(13,17,'                                                            ');
    putit(13,18,'                                                            ');
    putit(13,19,'     | Press RETURN to access the UMPIRE section |          ');
    putit(13,20,'     |              and to begin play            |          ');
    read(d)
  end;                              {scribble}

PROCEDURE CHECKIT;
  {Checks to see whether game is a resumption of a previous game.  If so,
   then boot program is terminated and an indicator is passed to the main
   program via a chain variable.}
  var
    d : string; dd : char;
  begin
    clearscreen;
    gotoxy(25,10);
    writeln('Is the game to be a resumption');
    gotoxy(25,11);
    write('of a previous play (Y or N): ');
```

```
    read(dd);
    if (dd='Y') or (dd='y') then
      begin
        SETCVAL('RES');
        disk2;
        exit(STARTUP)
      end
    else SETCVAL('NRES')
  end;                          (checkit)

begin
  clearscreen;draw1;draw2;
  setchain('/TEMPO/tempo');
  checkit;
  scribble;
  setnames;
  disk2;
  rewrite (datafile,'TURNKEY1:name.data');
  for i := 1 to 4 do
    begin
      datafile^ := component[i];
      put (datafile)
    end;
  close (datafile,lock)
end.                            (startup)
```

```
{$U+}
PROGRAM TEMPO;

  USES APPLESTUFF, CHAINSTUFF, MYSTUFF;
  {Main driver.}

  Type
    forecast = packed record
                 name : string;
                 wravail : integer;
                 wrRaOstarts : integer;
                 wrofRaO : integer ;
                 wr1RaOcost : integer;
                 wr2RaOcost : integer;
                 wr3RaOcost : integer;
                 inventory : integer;
                 tval : integer;
                 AQcost : integer;
                 OPcost : integer;
                 utils : integer;
                 purlimit : integer;
                 OPcostTTL : integer;
                 utilsTTL : integer;
                 delete : integer;
                 penalty : integer;
                 unitbuy : integer;
                 status : integer;
                 sort : string
               end;

    systems = packed array [1..30] of forecast;

    temp = packed record
             twar,tredleft,tblueleft,trawspent,tbaspent,trispent,tbispent,
               trrdspent,tbrdspent,tnumofsystems,troa,trob,trda,trdb,tboa,tbob,
               tbda,tbdb,tro,trd,trrdo,trrdd,tfro,tfrd,tfrrdo,tfrrdd,
               tbo,tbd,tbrdo,tbrdd,tfbo,tfbd,tfbrdo,tfbrdd : integer;
             tmtr,twarcont : char;
             tredbudget,tblubudget : array[1..20] of integer;
             tplayers : array[1..4] of string
           end;

  var
    ded : string;
    rs : systems;
    bs : systems;
    sys : systems;
    year : integer;
    redbudget, blubudget : array [1..20] of integer;
    players : array [1..4] of string;
    mainflag : boolean;
    redleft, blueleft : integer;
    rawspent, baspent : integer;
    rispent, bispent : integer;
    rrdspent, brdspent : integer;
    numofsystems : integer;
    mtr : char;
    z : text;
    roa,rob,rda,rdb,boa,bob,bda,bdb : integer;
    ro,rd,rrdo,rrdd,fro,frd,frrdo,frrdd : integer;
    bo,bd,brdo,brdd,fbo,fbd,fbrdo,fbrdd : integer;
```

```
  warcont : char;



{$INCLUDE NP2:BATCH1.text}
{$INCLUDE THESIS2:BATCH1A.text}
{$INCLUDE THESIS2:BATCH1B.text}
{$INCLUDE NP2:BATCH2.text}
{$INCLUDE THESIS2:BATCH2A.text}
{$INCLUDE THESIS2:BATCH2B.text}
{$INCLUDE THESIS2:BATCH3.text}



PROCEDURE MAINMENU (person : string);
  {The main menu.  Provides a central calling point from which all
   expenditure decisions originate, and from which all pertinent
   information obtained.}
  var
    selection : char; quitflag : boolean;
  begin
    quitflag := false;
    repeat
      clearscreen;
      putit(33,1,'MAIN MENU');
      putit(10,6,'1...STATUS OF FORCES CURRENTLY IN INVENTORY');
      putit(10,8,'2...STATUS OF FORCES AVAILABLE BUT NOT IN INVENTORY');
      putit(10,10,'3...CURRENT RESEARCH AND DEVELOPMENT CANDIDATES');
      putit(10,12,'4...INTELLIGENCE/COUNTERINTELLIGENCE INFORMATION');
      putit(10,14,'5...CURRENT BUDGETARY/SYSTEMS PURCHASE STATUS');
      putit(10,16,'6...QUIT (MAKE NO FURTHER BUDGET DECISIONS THIS YEAR)');
      writeln;writeln;writeln;
      write('                    Enter the desired option number: ');
      read(selection);
      case selection of
        '1' : opstatus(person);
        '2' : austatus(person);
        '3' : rdstatus(person);
        '4' : intelstatus(person);
        '5' : currentstatus(person);
        '6' : quitflag := true;
       otherwise  help
      end;
    until quitflag
  end;                              {mainmenu}

PROCEDURE STARTUP ( person : string );
  {Initializes the generic record sys for global use by players.}
  begin
    if person = players[1]
      then sys := rs
    else sys := bs;
  end;                          {startup}

PROCEDURE PLAYERONE;
  {Allows annual budget decisions for player 1.  Updates all pertinent
   variables for computation of winner and usage in following play.}
  begin
    startup (players[1]);
```

```
    update (players[1],redbudget[year]);
    mainmenu (players[1]);
    print (players[1])
  end;                          {playerone}

PROCEDURE PLAYERTWO;
  {See Player1 routine.}
  begin
    startup (players[2]);
    update (players[2],blubudget[year]);
    mainmenu (players[2]);
    print (players[2])
  end;                          {playertwo}

PROCEDURE GAMEEND;
  {Closes printers.}
  begin
    clearscreen;
    if NOT (prtr = '3') then close(z,lock)
  end;                          {gameend}



begin
  randomize;
  GETCURL(ded);
  if ded='RES' then setall
    else
      begin
        initialize;
        whoplays;
        umpire
      end;
  mainflag := false;
  repeat
      year := year + 1;
      initwr;
      playerone;
      playertwo;
      yearend
  until mainflag;
  gameend
end.                            {tempo}
```

```
(THE FOLLOWING SEGMENTED PROCEDURE IS COMPILED AND INSERTED IN THE MAIN
 PROGRAM AS BATCH 1 USING THE COMPILER 'INCLUDE' OPTION, DUE TO FILE
 SIZE CONSTRAINTS.)


   SEGMENT PROCEDURE INITIALIZE;
    (Initializes 17 weapon systems using default parameters;
     initializes budgets to default figures; initializes
     year, expenditure variables and number of systems)

    PROCEDURE INITOA;
      begin
        rs[1].name := 'OA1';rs[1].wravail := 1;rs[1].status := 0;
        rs[1].wrRaDstarts := 0;rs[1].wrofRaD := 0;
        rs[1].wr1RaDcost := 0;rs[1].wr2RaDcost := 0;
        rs[1].wr3RaDcost := 0;rs[1].inventory := 40;
        rs[1].AQcost := 50;rs[1].OPcost := 30;
        rs[1].utils := 20;rs[1].purlimit := 25;
        rs[1].unitbuy := 0;rs[1].sort := 'OA';
        rs[1].delete := 0;rs[1].penalty := 0;
        rs[1].OPcostTTL := (rs[1].OPcost)*(rs[1].inventory);
        rs[1].utilsTTL := (rs[1].utils)*(rs[1].inventory);
        rs[2].name := 'OA2';rs[2].wravail := 4;rs[2].status := 1;
        rs[2].wrRaDstarts := 1;rs[2].wrofRaD := 0;
        rs[2].wr1RaDcost := 400;rs[2].wr2RaDcost := 900;
        rs[2].wr3RaDcost := 700;rs[2].inventory := 0;
        rs[2].AQcost := 300;rs[2].OPcost := 175;
        rs[2].utils := 300;rs[2].purlimit := 15;
        rs[2].unitbuy := 0;rs[2].sort := 'OA';
        rs[2].delete := 0;rs[2].penalty := 0;
        rs[2].OPcostTTL := (rs[2].OPcost)*(rs[2].inventory);
        rs[2].utilsTTL := (rs[2].utils)*(rs[2].inventory);
        rs[3].name := 'OA3';rs[3].wravail := 6;rs[3].status := 1;
        rs[3].wrRaDstarts := 3;rs[3].wrofRaD := 0;
        rs[3].wr1RaDcost := 800;rs[3].wr2RaDcost := 1000;
        rs[3].wr3RaDcost := 600;rs[3].inventory := 0;
        rs[3].AQcost := 200;rs[3].OPcost := 100;
        rs[3].utils := 250;rs[3].purlimit := 15;
        rs[3].unitbuy := 0;rs[3].sort := 'OA';
        rs[3].delete := 0;rs[3].penalty := 0;
        rs[3].OPcostTTL := (rs[3].OPcost)*(rs[3].inventory);
        rs[3].utilsTTL := (rs[3].utils)*(rs[3].inventory)
      end;                              {initoa}

    PROCEDURE INITSET1OB;
      begin
        rs[4].name := 'OB1';rs[4].wravail := 1;rs[4].status := 0;
        rs[4].wrRaDstarts := 0;rs[4].wrofRaD := 0;
        rs[4].wr1RaDcost := 0;rs[4].wr2RaDcost := 0;
        rs[4].wr3RaDcost := 0;rs[4].inventory := 20;
        rs[4].AQcost := 70;rs[4].OPcost := 150;
        rs[4].utils := 120;rs[4].purlimit := 15;
        rs[4].unitbuy := 0;rs[4].sort := 'OB';
        rs[4].delete := 0;rs[4].penalty := 0;
        rs[4].OPcostTTL := (rs[4].OPcost)*(rs[4].inventory);
        rs[4].utilsTTL := (rs[4].utils)*(rs[4].inventory);
        rs[5].name := 'OB1mod';rs[5].wravail := 3;rs[5].status := 1;
        rs[5].wrRaDstarts := 2;rs[5].wrofRaD := 0;
        rs[5].wr1RaDcost := 100;rs[5].wr2RaDcost := 0;
```

```
    rs[5].wr3RaOcost := 0;rs[5].inventory := 0;
    rs[5].AQcost := 30;rs[5].OPcost := 120;
    rs[5].utils := 140;rs[5].purlimit := 15;
    rs[5].unitbuy := 0;rs[5].sort := '08';
    rs[5].delete := 0;rs[5].penalty := 0;
    rs[5].OPcostTTL := (rs[5].OPcost)*(rs[5].inventory);
    rs[5].utilsTTL := (rs[5].utils)*(rs[5].inventory);
    rs[6].name := '082';rs[6].wravail := 2;rs[6].status := 1;
    rs[6].wrRaOstarts := 1;rs[6].wrofRaO := 0;
    rs[6].wr1RaOcost := 600;rs[6].wr2RaOcost := 0;
    rs[6].wr3RaOcost := 0;rs[6].inventory := 0;
    rs[6].AQcost := 75;rs[6].OPcost := 35;
    rs[6].utils := 40;rs[6].purlimit := 35;
    rs[6].unitbuy := 0;rs[6].sort := '08';
    rs[6].delete := 0;rs[6].penalty := 0;
    rs[6].OPcostTTL := (rs[6].OPcost)*(rs[6].inventory);
    rs[6].utilsTTL := (rs[6].utils)*(rs[6].inventory)
  end;                          {initSET1ob}

PROCEDURE INITSET2OB;
  begin
    rs[7].name := '083';rs[7].wravail := 4;rs[7].status := 1;
    rs[7].wrRaOstarts := 2;rs[7].wrofRaO := 0;
    rs[7].wr1RaOcost := 500;rs[7].wr2RaOcost := 800;
    rs[7].wr3RaOcost := 0;rs[7].inventory := 0;
    rs[7].AQcost := 100;rs[7].OPcost := 50;
    rs[7].utils := 125;rs[7].purlimit := 15;
    rs[7].unitbuy := 0;rs[7].sort := '08';
    rs[7].delete := 0;rs[7].penalty := 0;
    rs[7].OPcostTTL := (rs[7].OPcost)*(rs[7].inventory);
    rs[7].utilsTTL := (rs[7].utils)*(rs[7].inventory);
    rs[8].name := '083mod';rs[8].wravail := 3;rs[8].status := 1;
    rs[8].wrRaOstarts := 2;rs[8].wrofRaO := 0;
    rs[8].wr1RaOcost := 600;rs[9].wr2RaOcost := 0;
    rs[8].wr3RaOcost := 0;rs[8].inventory := 0;
    rs[8].AQcost := 60;rs[8].OPcost := 75;
    rs[8].utils := 200;rs[8].purlimit := 15;
    rs[9].unitbuy := 0;rs[8].sort := '08';
    rs[8].delete := 0;rs[8].penalty := 0;
    rs[8].OPcostTTL := (rs[8].OPcost)*(rs[8].inventory);
    rs[8].utilsTTL := (rs[8].utils)*(rs[8].inventory);
    rs[9].name := '084';rs[9].wravail := 5;rs[9].status := 1;
    rs[9].wrRaOstarts := 3;rs[9].wrofRaO := 0;
    rs[9].wr1RaOcost := 500;rs[9].wr2RaOcost := 600;
    rs[9].wr3RaOcost := 0;rs[9].inventory := 0;
    rs[9].AQcost := 75;rs[9].OPcost := 25;
    rs[9].utils := 75;rs[9].purlimit := 30;
    rs[9].unitbuy := 0;rs[9].sort := '08';
    rs[9].delete := 0;rs[9].penalty := 0;
    rs[9].OPcostTTL := (rs[9].OPcost)*(rs[9].inventory);
    rs[9].utilsTTL := (rs[9].utils)*(rs[9].inventory)
  end;                          {initSET2ob}

PROCEDURE INITDA;

  PROCEDURE INITDAAUX;
    begin
      rs[10].name := 'DA1';rs[10].wravail := 1;rs[10].status := 0;
      rs[10].wrRaOstarts := 0;rs[10].wrofRaO := 0;
      rs[10].wr1RaOcost := 0;rs[10].wr2RaOcost := 0;
```

```
      rs[10].wr3RaOcost := 0;rs[10].inventory := 20;
      rs[10].AQcost := 100;rs[10].OPcost := 60;
      rs[10].utils := 50;rs[10].purlimit := 25;
      rs[10].unitbuy := 0;rs[10].sort := 'DA';
      rs[10].delete := 0;rs[10].penalty := 0;
      rs[10].OPcostTTL := (rs[10].OPcost)*(rs[10].inventory);
      rs[10].utilsTTL := (rs[10].utils)*(rs[10].inventory);
      rs[11].name := 'DA1mod';rs[11].wravail := 2;rs[11].status := 1;
    end;                          {initdaaux}

  begin
    initdaaux;
    rs[11].wrRaOstarts := 1;rs[11].wrofRaO := 0;
    rs[11].wr1RaOcost := 200;rs[11].wr2RaOcost := 0;
    rs[11].wr3RaOcost := 0;rs[11].inventory := 0;
    rs[11].AQcost := 50;rs[11].OPcost := 80;
    rs[11].utils := 80;rs[11].purlimit := 25;
    rs[11].unitbuy := 0;rs[11].sort := 'DA';
    rs[11].delete := 0;rs[11].penalty := 0;
    rs[11].OPcostTTL := (rs[11].OPcost)*(rs[11].inventory);
    rs[11].utilsTTL := (rs[11].utils)*(rs[11].inventory);
    rs[12].name := 'DA2';rs[12].wravail := 5;rs[12].status := 1;
    rs[12].wrRaOstarts := 2;rs[12].wrofRaO := 0;
    rs[12].wr1RaOcost := 300;rs[12].wr2RaOcost := 900;
    rs[12].wr3RaOcost := 900;rs[12].inventory := 0;
    rs[12].AQcost := 50;rs[12].OPcost := 40;
    rs[12].utils := 120;rs[12].purlimit := 20;
    rs[12].unitbuy := 0;rs[12].sort := 'DA';
    rs[12].delete := 0;rs[12].penalty := 0;
    rs[12].OPcostTTL := (rs[12].OPcost)*(rs[12].inventory);
    rs[12].utilsTTL := (rs[12].utils)*(rs[12].inventory);
    rs[13].name := 'DA3';rs[13].wravail := 5;rs[13].status := 1;
    rs[13].wrRaOstarts := 3;rs[13].wrofRaO := 0;
    rs[13].wr1RaOcost := 1000;rs[13].wr2RaOcost := 1500;
    rs[13].wr3RaOcost := 0;rs[13].inventory := 0;
    rs[13].AQcost := 60;rs[13].OPcost := 20;
    rs[13].utils := 90;rs[13].purlimit := 25;
    rs[13].unitbuy := 0;rs[13].sort := 'DA';
    rs[13].delete := 0;rs[13].penalty := 0;
    rs[13].OPcostTTL := (rs[13].OPcost)*(rs[13].inventory);
    rs[13].utilsTTL := (rs[13].utils)*(rs[13].inventory)
  end;                            {initda}

PROCEDURE INITDB;

  PROCEDURE INITDBAUX;
    begin
      rs[14].name := 'DB1';rs[14].wravail := 1;rs[14].status := 0;
      rs[14].wrRaOstarts := 0;rs[14].wrofRaO := 0;
      rs[14].wr1RaOcost := 0;rs[14].wr2RaOcost := 0;
      rs[14].wr3RaOcost := 0;rs[14].inventory := 100;
      rs[14].AQcost := 40;rs[14].OPcost := 20;
      rs[14].utils := 15;rs[14].purlimit := 25;
      rs[14].unitbuy := 0;rs[14].sort := 'DB';
      rs[14].delete := 0;rs[14].penalty := 0;
      rs[14].OPcostTTL := (rs[14].OPcost)*(rs[14].inventory);
      rs[14].utilsTTL := (rs[14].utils)*(rs[14].inventory)
    end;                          {initdbaux}

  begin
    initdbaux;
    rs[15].name := 'DB2';rs[15].wravail := 3;rs[15].status := 1;
```

```
    rs[15].wrRaOstarts := 1;rs[15].wrofRaO := 0;
    rs[15].wr1RaOcost := 400;rs[15].wr2RaOcost := 400;
    rs[15].wr3RaOcost := 0;rs[15].inventory := 0;
    rs[15].AQcost := 90;rs[15].OPcost := 70;
    rs[15].utils := 100;rs[15].purlimit := 25;
    rs[15].unitbuy := 0;rs[15].sort := 'OB';
    rs[15].delete := 0;rs[15].penalty := 0;
    rs[15].OPcostTTL := (rs[15].OPcost)*(rs[15].inventory);
    rs[15].utilsTTL := (rs[15].utils)*(rs[15].inventory);
    rs[16].name := 'OB3';rs[16].wravail := 3;rs[16].status := 1;
    rs[16].wrRaOstarts := 2;rs[16].wrofRaO := 0;
    rs[16].wr1RaOcost := 1000;rs[16].wr2RaOcost := 0;
    rs[16].wr3RaOcost := 0;rs[16].inventory := 0;
    rs[16].AQcost := 100;rs[16].OPcost := 50;
    rs[16].utils := 200;rs[16].purlimit := 25;
    rs[16].unitbuy := 0;rs[16].sort := 'OB';
    rs[16].delete := 0;rs[16].penalty := 0;
    rs[16].OPcostTTL := (rs[16].OPcost)*(rs[16].inventory);
    rs[16].utilsTTL := (rs[16].utils)*(rs[16].inventory);
    rs[17].name := 'OB4';rs[17].wravail := 5;rs[17].status := 1;
    rs[17].wrRaOstarts := 3;rs[17].wrofRaO := 0;
    rs[17].wr1RaOcost := 600;rs[17].wr2RaOcost := 600;
    rs[17].wr3RaOcost := 0;rs[17].inventory := 0;
    rs[17].AQcost := 150;rs[17].OPcost := 225;
    rs[17].utils := 400;rs[17].purlimit := 10;
    rs[17].unitbuy := 0;rs[17].sort := 'OB';
    rs[17].delete := 0;rs[17].penalty := 0;
    rs[17].OPcostTTL := (rs[17].OPcost)*(rs[17].inventory);
    rs[17].utilsTTL := (rs[17].utils)*(rs[17].inventory);
  end;                                {initdb}

PROCEDURE INITGENRLDATA;
  begin
    bs := rs;
    redbudset[1] := 9300;redbudset[2] := 9600;redbudset[3] := 9400;
    redbudset[4] := 9200;redbudset[5] := 9100;redbudset[6] := 9100;
    redbudset[7] := 9100;redbudset[8] := 9100;redbudset[9] := 9100;
    redbudset[10] := 9100;
    redbudset[11] := 9300;redbudset[12] := 9600;redbudset[13] := 9400;
    redbudset[14] := 9200;redbudset[15] := 9100;redbudset[16] := 9100;
    redbudset[17] := 9100;redbudset[18] := 9100;redbudset[19] := 9100;
    redbudset[20] := 9100;
    blubudset := redbudset;
    raespent := 0;
    baespent := 0;rispent := 0;bispent := 0;rrdspent := 0;ordspent := 0;
    numofsystems := 17;
    year := 0;warcont := 'N'
  end;                                {initgenrldata}

begin
  initoa;initsetlob;initset2ob;initda;initdb;initgenrldata
end;                                  {initialize}
```

```
{THE FOLLOWING SEGMENTED PROCEDURES ARE COMPILED AND INSERTED IN THE MAIN
 PROGRAM AS BATCH 1A USING THE COMPILER 'INCLUDE' OPTION, DUE TO FILE
 SIZE CONSTRAINTS.}


SEGMENT PROCEDURE INTELSTATUS(person : string);
     {Accessed from main menu.  Provides status of current intellisence.
      Allows the procurement of future intellisence, the display of
      acquired intellisence, and the scrapping of all requests budgeted
      for the current year.}
     var
       iflag : boolean; d : char;

     PROCEDURE RIDINFO;
       var
         e : char;

       PROCEDURE BLUERID;
         begin
           if (fbo = 100) or (fbo = -100) then
             begin
               fbo:=fbo-100; bispent:=bispent-100; bluleft:=bluleft+100
             end;
           if (fbd = 100) or (fbd = -100) then
             begin
               fbd:=fbd-100; bispent:=bispent-100; bluleft:=bluleft+100
             end;
           if (fbrdo = 100) or (fbrdo = -100) then
             begin
               fbrdo:=fbrdo-100; bispent:=bispent-100; bluleft:=bluleft+100
             end;
           if (fbrdd = 100) or (fbrdd = -100) then
             begin
               fbrdd:=fbrdd-100; bispent:=bispent-100; bluleft:=bluleft+100
             end;
           if (fro = -200) or (fro = -100) then
             begin
               fro:=fro+200; bispent:=bispent-200; bluleft:=bluleft+200
             end;
           if (frd = -200) or (frd = -100) then
             begin
               frd:=frd+200; bispent:=bispent-200; bluleft:=bluleft+200
             end;
           if (frrdo = -200) or (frrdo = -100) then
             begin
               frrdo:=frrdo+200; bispent:=bispent-200; bluleft:=bluleft+200
             end;
           if (frrdd = -200) or (frrdd = -100) then
             begin
               frrdd:=frrdd+200; bispent:=bispent-200; bluleft:=bluleft+200
             end
         end;                                  {blurerid}

       PROCEDURE REDRID;
         begin
           if (fro = 100) or (fro = -100) then
             begin
               fro:=fro-100; rispent:=rispent-100; redleft:=redleft+100
             end;
```

```
      if (frd = 100) or (frd = -100) then
        begin
          frd:=frd-100; rispent:=rispent-100; redleft:=redleft+100
        end;
      if (frrdo = 100) or (frrdo = -100) then
        begin
          frrdo:=frrdo-100; rispent:=rispent-100; redleft:=redleft+100
        end;
      if (frrdd = 100) or (frrdd = -100) then
        begin
          frrdd:=frrdd-100; rispent:=rispent-100; redleft:=redleft+100
        end;
      if (fbo = -200) or (fbo = -100) then
        begin
          fbo:=fbo+200; rispent:=rispent-200; redleft:=redleft+200
        end;
      if (fbd = -200) or (fbd = -100) then
        begin
          fbd:=fbd+200; rispent:=rispent-200; redleft:=redleft+200
        end;
      if (fbrdo = -200) or (fbrdo = -100) then
        begin
          fbrdo:=fbrdo+200; rispent:=rispent-200; redleft:=redleft+200
        end;
      if (fbrdd = -200) or (fbrdd = -100) then
        begin
          fbrdd:=fbrdd+200; rispent:=rispent-200; redleft:=redleft+200
        end
    end;                              {redrid}


  begin
    if person = players[1] then redrid
      else bluerid;
    clearscreen;
    putit(15,10,'ALL INTELLIGENCE/COUNTERINTELLIGENCE PROCUREMENTS');
    putit(15,11,'          HAVE BEEN CANCELLED FOR THE YEAR.');
    putit(15,13,'             <Press RETURN to Continue>');
    read(e)
  end;                                {ridinfo}

PROCEDURE BUYINFO;
  var
    e : char;

  PROCEDURE BUYMENU;
    var
      f : char;

    PROCEDURE DOITRED;

      PROCEDURE AA;
        begin
          if (fro=100) or (fro=-100) then
            begin
              clearscreen;
              putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
              putit(18,11,'          <Press RETURN to Continue>');
              read(d);
```

```
            exit(busmenu)
          end;
        fra:=fra+100; rispent:=rispent+100; redleft:=redleft-100;
        clearscreen;
        putit(12,10,'You have just alloted $100 for intellisence procurem
ent.');
        putit(12,11,'                    <Press RETURN to Continue>');
        read(d)
      end;                               {aa}


    PROCEDURE BB;
      begin
        if (frd=100) or (frd=-100) then
          begin
            clearscreen;
            putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
            putit(16,11,'           <Press RETURN to Continue>');
            read(d);
            exit(busmenu)
          end;
        frd:=frd+100; rispent:=rispent+100; redleft:=redleft-100;
        clearscreen;
        putit(12,10,'You have just alloted $100 for intellisence procurem
ent.');
        putit(12,11,'                    <Press RETURN to Continue>');
        read(d)
      end;                               {bb}


    PROCEDURE CC;
      begin
        if (frrdo=100) or (frrdo=-100) then
          begin
            clearscreen;
            putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
            putit(16,11,'           <Press RETURN to Continue>');
            read(d);
            exit(busmenu)
          end;
        frrdo:=frrdo+100; rispent:=rispent+100; redleft:=redleft-100;
        clearscreen;
        putit(12,10,'You have just alloted $100 for intellisence procurem
ent.');
        putit(12,11,'                    <Press RETURN to Continue>');
        read(d)
      end;                               {cc}


    PROCEDURE DD;
      begin
        if (frrdd=100) or (frrdd=-100) then
          begin
            clearscreen;
            putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
            putit(16,11,'           <Press RETURN to Continue>');
```

```
                read(d);
                exit(buswenu)
              end;
            frrda:=frrda+100; rispent:=rispent+100; redleft:=redleft-100;
            clearscreen;
            putit(12,10,'You have just alloted $100 for intellisence procurem
ent.');
            putit(12,11,'                    <Press RETURN to Continue>');
            read(d)
          end;                               {dd}

        PROCEDURE EE;
          begin
            if (fbo=-200) or (fbo=-100) then
              begin
                clearscreen;
                putit(10,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                putit(10,11,'           <Press RETURN to Continue>');
                read(d);
                exit(buswenu)
              end;
            fbo:=fbo-200; rispent:=rispent+200; redleft:=redleft-200;
            clearscreen;
            putit(5,10,'You have just alloted $200 for intellisence/counterin
tellisence procurement.');
            putit(15,11,'                    <Press RETURN to Continue>');
            read(d)
          end;                               {ee}

        PROCEDURE FF;
          begin
            if (fbd=-200) or (fbd=-100) then
              begin
                clearscreen;
                putit(10,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                putit(10,11,'           <Press RETURN to Continue>');
                read(d);
                exit(buswenu)
              end;
            fbd:=fbd-200; rispent:=rispent+200; redleft:=redleft-200;
            clearscreen;
            putit(5,10,'You have just alloted $200 for intellisence/counterin
tellisence procurement.');
            putit(15,11,'                    <Press RETURN to Continue>');
            read(d)
          end;                               {ff}

        PROCEDURE GG;
          begin
            if (fbrdo=-200) or (fbrmo=-100) then
              begin
                clearscreen;
                putit(10,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                putit(10,11,'           <Press RETURN to Continue>');
                read(d);
                exit(buswenu)
              end;
```

```
              fbrdd:=fbrdd-200; risent:=risent+200; redleft:=redleft-200;
              clearscreen;
              putit(5,10,'You have just alloted $200 for intellisence/counterin
tellisence procurement.');
              putit(15,11,'                    <Press RETURN to Continue>');
              read(d)
            end;                               {aa}

          PROCEDURE HH;
            begin
              if (fbrdd=-200) or (fbrdd=-100) then
                begin
                  clearscreen;
                  putit(10,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                  putit(10,11,'           <Press RETURN to Continue>');
                  read(d);
                  exit(buymenu)
                end;
              fbrdd:=fbrdd-200; risent:=risent+200; redleft:=redleft-200;
              clearscreen;
              putit(5,10,'You have just alloted $200 for intellisence/counterin
tellisence procurement.');
              putit(15,11,'                    <Press RETURN to Continue>');
              read(d)
            end;                               {hh}


          begin
            case f of
              'a','A' : AA;
              'b','B' : BB;
              'c','C' : CC;
              'd','D' : DD;
              'e','E' : EE;
              'f','F' : FF;
              'g','G' : GG;
              'h','H' : HH;
              'q','Q' : iflag := true
            end
          end;                          {doitred}


        PROCEDURE DOITBLUE;

          PROCEDURE AA;
            begin
              if (fbo=100) or (fbo=-100) then
                begin
                  clearscreen;
                  putit(10,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                  putit(10,11,'           <Press RETURN to Continue>');
                  read(d);
                  exit(buymenu)
                end;
              fbo:=fbo+100; bisent:=bisent+100; bluleft:=bluleft-100;
              clearscreen;
              putit(12,10,'You have just alloted $100 for intellisence procurem
ent.');
```

```
      putit(12,11,'                    <Press RETURN to Continue>');
      read(d)
    end;                               {aa}


  PROCEDURE BB;
    begin
      if (fbd=100) or (fbd=-100) then
        begin
          clearscreen;
          putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
          putit(16,11,'              <Press RETURN to Continue>');
          read(d);
          exit(buymenu)
        end;
      fbd:=fbd+100; bispent:=bispent+100; bluleft:=bluleft-100;
      clearscreen;
      putit(12,10,'You have just alloted $100 for intellisence procurem
ent.');
      putit(12,11,'                    <Press RETURN to Continue>');
      read(d)
    end;                               {bb}


  PROCEDURE CC;
    begin
      if (fbrdo=100) or (fbrdo=-100) then
        begin
          clearscreen;
          putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
          putit(16,11,'              <Press RETURN to Continue>');
          read(d);
          exit(buymenu)
        end;
      fbrdo:=fbrdo+100; bispent:=bispent+100; bluleft:=bluleft-100;
      clearscreen;
      putit(12,10,'You have just alloted $100 for intellisence procurem
ent.');
      putit(12,11,'                    <Press RETURN to Continue>');
      read(d)
    end;                               {cc}


  PROCEDURE DD;
    begin
      if (fbrdd=100) or (fbrdd=-100) then
        begin
          clearscreen;
          putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
          putit(16,11,'              <Press RETURN to Continue>');
          read(d);
          exit(buymenu)
        end;
      fbrdd:=fbrdd+100; bispent:=bispent+100; bluleft:=bluleft-100;
      clearscreen;
      putit(12,10,'You have just alloted $100 for intellisence procurem
```

```
ent.');
                putit(12,11,'                    <Press RETURN to Continue>');
                read(d)
              end;                               {dd}

            PROCEDURE EE;
              begin
                if (fro=200) or (fro=100) then
                  begin
                    clearscreen;
                    putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                    putit(16,11,'           <Press RETURN to Continue>');
                    read(d);
                    exit(buymenu)
                  end;
                fro:=fro-200; bispent:=bispent+200; bluleft:=bluleft-200;
                clearscreen;
                putit(5,10,'You have just allated $200 for intellisence/counterin
tellisence procurement.');
                putit(15,11,'                    <Press RETURN to Continue>');
                read(d)
              end;                               {ee}

            PROCEDURE FF;
              begin
                if (frd=200) or (frd=100) then
                  begin
                    clearscreen;
                    putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                    putit(16,11,'           <Press RETURN to Continue>');
                    read(d);
                    exit(buymenu)
                  end;
                frd:=frd-200; bispent:=bispent+200; bluleft:=bluleft-200;
                clearscreen;
                putit(5,10,'You have just alloted $200 for intellisence/counterin
tellisence procurement.');
                putit(15,11,'                    <Press RETURN to Continue>');
                read(d)
              end;                               {ff}

            PROCEDURE GG;
              begin
                if (frrdo=200) or (frrdo=100) then
                  begin
                    clearscreen;
                    putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                    putit(16,11,'           <Press RETURN to Continue>');
                    read(d);
                    exit(buymenu)
                  end;
                frrdo:=frrdo-200; bispent:=bispent+200; bluleft:=bluleft-200;
                clearscreen;
                putit(5,10,'You have just alloted $200 for intellisence/counterin
tellisence procurement.');
                putit(15,11,'                    <Press RETURN to Continue>');
```

```
            read(d)
          end;                                  {ee}

        PROCEDURE HH;
          begin
            if (frrdd=-200) or (frrdd=-100) then
              begin
                clearscreen;
                putit(16,10,'FUNDING FOR THIS ITEM HAS ALREADY BEEN PROVIDED'
);
                putit(16,11,'           <Press RETURN to Continue>');
                read(d);
                exit(buymenu)
              end;
            frrdd:=frrdd-200; bispent:=bispent+200; bluleft:=bluleft-200;
            clearscreen;
            putit(5,10,'You have just alloted $200 for intellisence/counterin
tellisence procurement.');
            putit(15,11,'                <Press RETURN to Continue>');
            read(d)
          end;                                  {HH}


        begin
          case f of
            'a','A' : AA;
            'b','B' : BB;
            'c','C' : CC;
            'd','D' : DD;
            'e','E' : EE;
            'f','F' : FF;
            'g','G' : GG;
            'h','H' : HH;
            'q','Q' : iflag := true
          end
        end;                                {doitblue}


        begin
          clearscreen;
          putit(32,1,'PURCHASE OPTIONS');
          putit(16,3,'A...Current Force Strength of Enemy Offensive Forces');
          putit(16,4,'B...Current Force Strength of Enemy Defensive Forces');
          putit(16,5,'C...Current Offensive Enemy R&D Projects');
          putit(16,6,'D...Current Defensive Enemy R&D Projects');
          putit(16,7,'E...Supply Counterintellisence Regarding Own Offensive
Forces');
          putit(16,8,'F...Supply Counterintellisence Regarding Own Defensive
Forces');
          putit(16,9,'G...Supply Counterintellisence Regarding Own Offensive
R&D');
          putit(16,10,'H...Supply Counterintellisence Regarding Own Defensive
 R&D');
          putit(16,11,'Q...Quit');
          putit(18,14,'Intellisence Cost per Item (A thru D): $100');
          putit(18,15,'Counterintellisence Cost per Item (E thru H): $200');
          if person = players[1] then
            begin
              gotoxy(18,16);writeln('Annual Defense Budget Remaining: $',redl
```

```
eft)
              end
            else
              begin
                gotoxy(18,18);writeln('Annual Defense Budget Remaining: $',blul
eft)
              end;
            putit(18,19,'ENTER OPTION LETTER: ');
            gotoxy(37,19);read(f);
            if person = players[1] then doitred
              else doitblue
          end;                                  {buymenu}


      begin
        repeat
          buymenu;
        until iflag
      end;                          {buyinfo}

    PROCEDURE GETINFO;
      var
        e : char;p,q,op,oq,o,i,c : integer;

      PROCEDURE LINE1;
        begin
          if person = players[1] then
            begin
              if no = -100 then
                begin
                  putit(10,9,'CIA unable to breach enemy counterintellisence ba
rrier');
                  exit(line1)
                end;
              if no = 100 then
                begin
                  p := random mod 200; q := random mod 200;
                  o := boa+bob; op := o-p; oq := o+q;
                  gotoxy(10,9);
                  writeln('Total enemy offensive utils are estimated to be betw
een ',op,' and ',oq);
                  exit(line1)
                end;
              putit(10,9,'     NO INFORMATION PROCURED')
            end
          else
            begin
              if bo = -100 then
                begin
                  putit(10,9,'CIA unable to breach enemy counterintellisence ba
rrier');
                  exit(line1)
                end;
              if bo = 100 then
                begin
                  p := random mod 200; q := random mod 200;
                  o := roa+rob; op := o-p; oq := o+q;
                  gotoxy(10,9);
                  writeln('Total enemy offensive utils are estimated to be betw
een ',op,' and ',oq);
```

```
                exit(line1)
              end;
            putit(10,9,'     NO INFORMATION PROCURED')
          end
      end;                          {line1}

    PROCEDURE LINE2;
      begin
        if person = players[1] then
          begin
            if rd = -100 then
              begin
                putit(10,12,'CIA unable to breach enemy counterintellisence b
arrier');
                exit(line2)
              end;
            if rd = 100 then
              begin
                p := random mod 200; a := random mod 200;
                o := bda+bdb; op := o-a; oa := o+a;
                gotoxy(10,12);
                writeln('Total enemy defensive utils are estimated to be betw
een ',op,' and ',oa);
                exit(line2)
              end;
            putit(10,12,'     NO INFORMATION PROCURED')
          end
        else
          begin
            if bd = -100 then
              begin
                putit(10,12,'CIA unable to breach enemy counterintellisence b
arrier');
                exit(line2)
              end;
            if bd = 100 then
              begin
                p := random mod 200; a := random mod 200;
                o := rda+rdb; op := o-a; oa := o+a;
                gotoxy(10,12);
                writeln('Total enemy defensive utils are estimated to be betw
een ',op,' and ',oa);
                exit(line2)
              end;
            putit(10,12,'     NO INFORMATION PROCURED')
          end
      end;                          {line2}

    PROCEDURE LINE3;
      begin
        c := 0;
        if person = players[1] then
          begin
            if rrdo = -100 then
              begin
                putit(10,15,'CIA unable to breach enemy counterintellisence b
arrier');
                exit(line3)
              end;
            if rrdo = 100 then
```

```
                  begin
                    for i:= 1 to numofsystems do
                      if (bs[i].status=2) and (bs[i].delete=0) then
                        if(bs[i].sort='OA') or (bs[i].sort='OB') then
                          c:=c+1;
                    gotoxy(10,15);
                    writeln('Reports indicate enemy currently funding ',c,' offen
sive R&D projects');
                    exit(line3)
                  end;
                putit(10,15,'     NO INFORMATION PROCURED')
              end
            else
              begin
                if brdo = -100 then
                  begin
                    putit(10,15,'CIA unable to breach enemy counterintellisence b
arrier');
                    exit(line3)
                  end;
                if brdo = 100 then
                  begin
                    for i:= 1 to numofsystems do
                      if (rs[i].status=2) and (rs[i].delete=0) then
                        if(rs[i].sort='OA') or (rs[i].sort='OB') then
                          c:=c+1;
                    gotoxy(10,15);
                    writeln('Reports indicate enemy currently funding ',c,' offen
sive R&D projects');
                    exit(line3)
                  end;
                putit(10,15,'     NO INFORMATION PROCURED')
              end
          end;                          (line3)


        PROCEDURE LINE4;
          begin
            c := 0;
            if person = players[1] then
              begin
                if rrdd = -100 then
                  begin
                    putit(10,18,'CIA unable to breach enemy counterintellisence b
arrier');
                    exit(line4)
                  end;
                if rrdd = 100 then
                  begin
                    for i:= 1 to numofsystems do
                      if (bs[i].status=2) and (bs[i].delete=0) then
                        if(bs[i].sort='OA') or (bs[i].sort='OB') then
                          c:=c+1;
                    gotoxy(10,18);
                    writeln('Reports indicate enemy currently funding ',c,' defen
sive R&D projects');
                    exit(line4)
                  end;
                putit(10,18,'     NO INFORMATION PROCURED')
```

```
            end
          else
            begin
              if brdd = -100 then
                begin
                  putit(10,18,'CIA unable to breach enemy counterintellisence b
arrier');
                  exit(line4)
                end;
              if brdd = 100 then
                begin
                  for i:= 1 to numofsystems do
                    if (rs[i].status=2) and (rs[i].delete=0) then
                      if(rs[i].sort='DA') or (rs[i].sort='DB') then
                      c:=c+1;
                  gotoxy(10,18);
                  writeln('Reports indicate enemy currently funding ',c,' defen
sive R&D projects');
                  exit(line4)
                end;
              putit(10,18,'     NO INFORMATION PROCURED')
            end
        end;                              {line4}



      begin
        clearscreen;
        putit(26,3,'CURRENT INTELLIGENCE REPORTS:');
        putit(5,8,'ENEMY OFFENSIVE FORCES INFORMATION:');
        line1;
        putit(5,11,'ENEMY DEFENSIVE FORCES INFORMATION:');
        line2;
        putit(5,14,'ENEMY OFFENSIVE R&D INFORMATION:');
        line3;
        putit(5,17,'ENEMY DEFENSIVE R&D INFORMATION:');
        line4;
        putit(27,21,'(Press RETURN to Continue)');
        read(e)
      end;                              {setinfo}

    begin
      iflag := false;
      repeat
        clearscreen;
        putit(3,5,'SELECT INTELLIGENCE OPTION: ');
        putit(13,9,'A...Receive Previously Purchased Intelligence Information'
);
        putit(13,11,'B...Purchase Intelligence/Counterintellisence Information'
);
        putit(13,13,'C...Cancel All Intelligence/Counterintellisence Requests'
);
        putit(13,14,'                              For the Current Year');
        putit(13,16,'Q...Quit');
        gotoxy(37,5);read(d);
        case d of
          'a','A' : setinfo;
          'b','B' : buyinfo;
          'c','C' : oldinfo;
          'q','Q' : iflag := true;
          otherwise help
        end;
      until iflag
    end;                              {intelstatus}
```

```
SEGMENT PROCEDURE INITYR;

 {Called annually.  Updates intellisence and expenditure variables.
  Computes projected total systems operating costs for upcoming year.
  Shelves undecided new R&D systems from previous year, and determines
  shelving penalties.  Decrements new budget by twice previous budget
  overrun.  Introduces random draw to adjust expected budget and R&D
  figures.}
 var
   jj,i,q,uniform,deviation : integer;

  FUNCTION MINIMUM : integer;
    begin
      if q <= 300 then minimum := q
        else minimum := 300
    end;                    {minimum}

  PROCEDURE RINITRD;
   var
     j, earlier : integer;

   PROCEDURE ASSUME;
     begin
       if rs[j].status = 2 then
         begin
           if rs[j].yrofRaD = 2 then
             begin
               rs[j].yr3RaDcost:=rs[j].yr3RaDcost+(50*jj);
               rndspent := rndspent + rs[j].yr3RaDcost;
               redleft := redleft - rs[j].yr3RaDcost
             end;
           if rs[j].yrofRaD = 1 then
             begin
               rs[j].yr2RaDcost:=rs[j].yr2RaDcost+(50*jj);
               rndspent := rndspent + rs[j].yr2RaDcost;
               redleft := redleft - rs[j].yr2RaDcost
             end
         end
     end;                       {assume}

   begin
     earlier := year - 1;
     for j := 1 to numofsystems do
       begin
         if (rs[j].yrRaDstarts <= earlier) and (rs[j].status = 1) then
           if rs[j].delete = 0 then
             begin
               rs[j].status := 3;
               rs[j].yravail := rs[j].yravail + 1
             end;
         if (rs[j].yrRaDstarts <= earlier) and (rs[j].status = 3) then
             if rs[j].delete = 0 then
               begin
                 rs[j].yravail := rs[j].yravail + 1;
                 if rs[j].yrofRaD = 0 then rs[j].penalty := 0;
                 if rs[j].yrofRaD = 1 then
                   begin
                     q := rs[j].yr1RaDcost;
                     rs[j].penalty := minimum
                   end;
                 if rs[j].yrofRaD = 2 then
                   begin
                     q := rs[j].yr2RaDcost;
                     rs[j].penalty := minimum
                   end
```

```
            end;
          if (rs[j].yrRaOstarts <= earlier) and (rs[j].status = 2) then
            if rs[j].delete = 0 then
              begin
                rs[j].ynofRaO := rs[j].yrofRaO +1;
                if rs[j].yravail = year then rs[j].status := 0;
                assume
              end
        end
    end;                              {reinitrd}

PROCEDURE BINITRD;
  var
    j, earlier : integer;

  PROCEDURE ASSUME;
    begin
      if bs[j].status = 2 then
        begin
          if bs[j].yrofRaO = 2 then
            begin
              bs[j].yr3RaOcost:=bs[j].yr3RaOcost+(50*jj);
              brdspent := brdspent + bs[j].yr3RaOcost;
              bluleft := bluleft - bs[j].yr3RaOcost
            end;
          if bs[j].yrofRaO = 1 then
            begin
              bs[j].yr2RaOcost:=bs[j].yr2RaOcost+(50*jj);
              brdspent := brdspent + bs[j].yr2RaOcost;
              bluleft := bluleft - bs[j].yr2RaOcost
            end
        end
    end;                              {assume}

  begin
    earlier := year - 1;
    for j := 1 to numofsystems do
      begin
        if (bs[j].yrRaOstarts <= earlier) and (bs[j].status = 1) then
          if bs[j].delete = 0 then
            begin
              bs[j].status := 3;
              bs[j].yravail := bs[j].yravail + 1
            end;
        if (bs[j].yrRaOstarts <= earlier) and (bs[j].status = 3) then
            if bs[j].delete = 0 then
              begin
                bs[j].yravail := bs[j].yravail + 1;
                if bs[j].yrofRaO = 0 then bs[j].penalty := 0;
                if bs[j].yrofRaO = 1 then
                  begin
                    q := bs[j].yr1RaOcost;
                    bs[j].penalty := minimum
                  end;
                if bs[j].yrofRaO = 2 then
                  begin
                    q := bs[j].yr2RaOcost;
                    bs[j].penalty := minimum
                  end
```

```
            end;
          if (bs[j].yrRaDstart <= earlier) and (bs[j].status = 2) then
            if bs[j].delete = 0 then
              begin
                bs[j].yrofRaD := bs[j].yrofRaD +1;
                if bs[j].yravail = year then bs[j].status := 0;
                assume
              end
        end
    end;                              {binitrd}

PROCEDURE INITINTEL;
  begin
    if year = 1 then
      begin
        fro := 0; fbo := 0;
        frd := 0; fbd := 0;
        frrdo := 0; fbrdo := 0;
        frrdd := 0; fbrdd := 0
      end;
    ro := fro; bo := fbo;
    rd := frd; bd := fbd;
    rrdo := frrdo; brdo := fbrdo;
    rrdd := frrdd; brdd := fbrdd;
    fro := 0; fbo := 0;
    frd := 0; fbd := 0;
    frrdo := 0; fbrdo := 0;
    frrdd := 0; fbrdd := 0
  end;                              {initintel}

begin
  if not (year = 1 ) then
    begin
      if (redleft < 0) then redbudget[year] := redbudget[year]+2*redleft;
      if (bluleft < 0) then blubudget[year] := blubudget[year]+2*bluleft;
      uniform := random mod 11;
      deviation := 150*uniform - 1000;
      redbudget[year] := redbudget[year] + deviation;
      blubudget[year] := blubudget[year] + deviation
    end;
  redleft := redbudget[year];bluleft := blubudget[year];
  rasspent := 0;baspent := 0;
  rispent := 0;bispent := 0;
  rrdspent := 0;brdspent := 0;
  for i := 1  to numofsystems do
    begin
      rs[i].unitbuy := rs[i].purlimit;
      rs[i].tval := rs[i].inventory;
      bs[i].unitbuy := bs[i].purlimit;
      bs[i].tval := bs[i].inventory;
      if (rs[i].inventory > 0) and (rs[i].delete = 0) then
        redleft := redleft - (rs[i].inventory*rs[i].opcost);
      if (bs[i].inventory > 0) and (bs[i].delete = 0) then
        bluleft := bluleft - (bs[i].inventory*bs[i].opcost)
    end;
  jj:= random mod 4;
  rinitrd; binitrd;
  initintel
end;                              {inityr}
```

```
{THE FOLLOWING SEGMENTED PROCEDURES ARE COMPILED AND INSERTED IN THE MAIN
 PROGRAM AS BATCH 1B USING THE COMPILER 'INCLUDE' OPTION, DUE TO FILE
 SIZE CONSTRAINTS.}


 SEGMENT PROCEDURE PRINT (person : string );
  {Called optionally.  Prints end-of-year decision summary for players.}
  var
    i,j,left,assgnmt,ispent,rdspent : integer;sysbudget : array [1..20] of integ
er;
    q : char;

  PROCEDURE TURNEND;
  var
    l : integer;
  begin
    roa := 0; boa := 0;
    rob := 0; bob := 0;
    rda := 0; bda := 0;
    rdb := 0; bdb := 0;
    for l := 1 to numofsystems do
      begin
        if (rs[l].sort = 'OA') and (rs[l].delete = 0) then
          roa := roa + rs[l].utilsttl;
        if (rs[l].sort = 'OB') and (rs[l].delete = 0) then
          rob := rob + rs[l].utilsttl;
        if (rs[l].sort = 'DA') and (rs[l].delete = 0) then
          rda := rda + rs[l].utilsttl;
        if (rs[l].sort = 'DB') and (rs[l].delete = 0) then
          rdb := rdb + rs[l].utilsttl;
        if (bs[l].sort = 'OA') and (bs[l].delete = 0) then
          boa := boa + bs[l].utilsttl;
        if (bs[l].sort = 'OB') and (bs[l].delete = 0) then
          bob := bob + bs[l].utilsttl;
        if (bs[l].sort = 'DA') and (bs[l].delete = 0) then
          bda := bda + bs[l].utilsttl;
        if (bs[l].sort = 'DB') and (bs[l].delete = 0) then
          bdb := bdb + bs[l].utilsttl
      end
  end;                          {turnend}

  PROCEDURE OUTPUT;

    PROCEDURE TITLE;
      begin
        writeln(z,' ');writeln(z,' ');writeln(z,' ');
        writeln(z,' ');writeln(z,' ');writeln(z,' ');
        writeln(z,'                                        ------------------------
');
        writeln(z,'                                       |                        |
');
        writeln(z,'                                       |  ANNUAL DECISION SUMMARY |
');
        writeln(z,'                                       |________________________|
');
        writeln(z,' ');
        writeln(z,'                                         PLAYER: ',person,     YEAR: ',yea
r);
        writeln(z,' ');writeln(z,' ');writeln(z,' ')
      end;                                  {title}

    PROCEDURE LIST1(n : integer);
      begin
        writeln(z,'            Name: Type : ',sys[n].name,' ',sys[n].sort,' ');
```

```
        writeln(z,'                    Inventory: ',sys[n].inventory);
        writeln(z,'                    System operation cost: ',sys[n].opcost,'
System acquisition cost: ',sys[n].acost);
        writeln(z,'                    Util value: ',sys[n].utils,'     Purchase rat
e: ',sys[n].purlimit);
        writeln(z,'                    Total operation costs: ',sys[n].OPcostTTL,'
   Total util value: ',sys[n].utilsTTL);
        writeln(z,' ');writeln(z,' ')
      end;                              {list1}

    PROCEDURE LIST2(n : integer);
      var
       m,mm : integer;
      begin
        writeln(z,'          Name(Type): ',sys[n].name,'(',sys[n].sort,')');
        mm:= sys[n].yrnfR&D;
        if sys[n].yr3R&Dcost>0 then m:=3
          else if sys[n].yr2R&Dcost>0 then m:=2
            else m:=1;
        writeln(z,'                    Yrs R&D completed: ',mm,'     Yrs of R&D requ
ired: ',m);
        writeln(z,'                    Costs:      Yr1: ',sys[n].yr1R&Dcost,'   Yr2
: ',sys[n].yr2R&Dcost,'   Yr3: ',sys[n].yr3R&Dcost);
        writeln(z,'                    System operation cost: ',sys[n].opcost,'
System acquisition cost: ',sys[n].acost);
        writeln(z,'                    Util value: ',sys[n].utils,'     Purchase rat
e: ',sys[n].purlimit);
        writeln(z,' ');writeln(z,' ')
      end;                              {list2}

    PROCEDURE INV;
      var
        n,count : integer;
      begin
        writeln(z,'     I.    SYSTEMS CURRENTLY IN INVENTORY:');
        writeln(z,' ');
        count:=0;
        for n:= 1 to numofsystems do
          begin
            if (sys[n].inventory>0) and (sys[n].delete=0)then
              begin
                list1(n);
                count:= count + 1
              end
          end;
        if count=0 then
          begin
            writeln(z,'                                        NONE');
            writeln(z,' ');writeln(z,' ')
          end
      end;                              {inv}


    PROCEDURE NOTINV;
      var
        n,count : integer;
      begin
        writeln(z,'     II.   SYSTEMS AVAILABLE BUT NOT CURRENTLY IN INVENTORY:'
);
```

```
    writeln(z,' ');
    count:=0;
    for n:= 1 to numofsystems do
      begin
        if (sys[n].inventory=0) and (sys[n].delete=0)then
          if sys[n].yravail<=year then
            begin
              list1(n);
              count:= count + 1
            end
      end;
    if count=0 then
      begin
        writeln(z,'                                    NONE');
        writeln(z,' ');writeln(z,' ')
      end
  end;                          {notim}

PROCEDURE TOTS;
  begin
    writeln(z,'     III.  TOTAL UTIL POINTS ACCUMULATED:');
    writeln(z,' ');
    if person=players[1] then
      begin
        writeln(z,'                    System type 0A: ',roa);
        writeln(z,'                    System type 0B: ',rob);
        writeln(z,'                    System type 0A: ',rda);
        writeln(z,'                    System type 0B: ',rdb)
      end
    else
      begin
        writeln(z,'                    System type 0A: ',boa);
        writeln(z,'                    System type 0B: ',bob);
        writeln(z,'                    System type 0A: ',bda);
        writeln(z,'                    System type 0B: ',bdb)
      end;
    writeln(z,' ');writeln(z,' ')
  end;                              {tots}

PROCEDURE CURRD;
  var
    n,count : integer;
  begin
    writeln(z,'     IV.   ONGOING R&D PROJECTS:');
    writeln(z,' ');
    count:=0;
    for n:= 1 to numofsystems do
      begin
        if (sys[n].status=2) and (sys[n].delete=0)then
          if sys[n].yrRsDstarts<=year then
            begin
              list2(n);
              count:= count + 1
            end
      end;
    if count=0 then
      begin
        writeln(z,'                                    NONE');
        writeln(z,' ');writeln(z,' ')
```

```
      end
    end;                              {summ}

  PROCEDURE SHURD;
    var
      n,count : integer;
    begin
      writeln(z,'     U.   SHELVED R&D PROJECTS:');
      writeln(z,' ');
      count:=0;
      for n:= 1 to numofsystems do
        begin
          if (sys[n].status=1) or (sys[n].status=3) then
            if (sys[n].yrRaDstarts<=year) and (sys[n].delete=0) then
              begin
                list2(n);
                count:= count + 1
              end
        end;
       if count=0 then
        begin
          writeln(z,'                                        NONE');
          writeln(z,' ');writeln(z,' ')
        end
    end;                              {shurd}

  PROCEDURE BUDG;
    var
      n : integer;

     PROCEDURE COSTS;
       var
         ij : integer;
       begin
         for ij := 1 to numofsystems do
           begin
             if (sys[ij].opcosttl > 0) and (sys[ij].yrsusl <= year)  then
               if sys[ij].delete = 0 then
                 begin
                   writeln(z,'                          Name(type): ',sys[ij].name,'
(',sys[ij].sort,')');
                   writeln(z,'                               Total operating cost: $
',sys[ij].opcosttl)
                 end
           end
       end;                           {costs}

      begin
        writeln(z,'     VI.  END-OF-YEAR BUDGET INFORMATION');
        writeln(z,' ');
        writeln(z,'          Total Annual Allowance: $',sysbudget[year]);
        writeln(z,'          Amount spent in acquisitions.................
.....$',aespent);
        writeln(z,'          Amount spent on intelligence.................
.....$',ispent);
        writeln(z,'          Amount spent on R and D.....................
.....$',rdspent);
        writeln(z,'          Operating costs by system (those in inventory
):');
```

```
        writeln(z,' ');
        costs;
        m := year + 1;
        writeln(z,' ');
        writeln(z,'                    Total monies remaining: $',left);
        writeln(z,'                    Expected defense budget for next year: $',sys
budget[m])
      end;                                        {budg}

    begin
      title;
      inv;
      notinv;
      tots;
      curRD;
      shuRD;
      budg
    end;                          {output}

  begin
    turnend;
    if person = players[1] then
      begin
        sysbudget := redbudget;left := redleft;sys := rs;
        aespent := raespent;ispent := rispent;rdspent := rrdspent
      end
    else
      begin
        sysbudget := blubudget;left := bluleft;sys := bs;
        aespent := baespent;ispent := bispent;rdspent := brdspent
      end;
    clearscreen;
    gotoxy(18,8);
    writeln('DECISIONS BY ',person,' FOR YEAR ',year,' ARE COMPLETED.');
    if NOT (prtr='3') then
      begin
        putit(15,10,'(Press ''P'' to obtain printout of year''s decisions)');
        putit(15,12,'                                or')
      end;
    putit(22,14,'(Press RETURN to continue game play)');
    read(d);
    if (d='p') or (d='P') then
      begin
        output;
        putit(14,10,'
');
        putit(14,12,'
');
        read(d)
      end
  end;                                {print}


SEGMENT PROCEDURE YEAREND;
  {Establishes annual game winner.  Computes the probability of war and
   either displays advisory to the umpire for his decision, or makes a
   random draw to compare against the computed probability to decide
   war event occurrence.  Creates data file on temp diskette if
   players decide to quit and resume game at a later time.}
  var
```

```
    red,blue : integer;

  PROCEDURE WINNER;
    var
      best : string; r1,r2,b1,b2 : integer; d : char;

    PROCEDURE DECLARE;
      begin
        clearscreen;
        putit(15,3,'|_____________________________________________|');
        putit(15,4,'|                                             |');
        putit(15,5,'|                                             |');
        putit(15,6,'|                                             |');
        putit(15,7,'|            AS A RESULT OF A TALLY OF        |');
        putit(15,8,'|            OPERATIONAL FORCES UTILITY       |');
        putit(15,9,'|            POINTS...                        |');
        putit(15,10,'|                                            |');
gotoxy(15,14);writeln('|              FOR GAME YEAR ',year,'.');putit(65,14,'|');
        putit(15,15,'|                                            |');
        putit(15,16,'|                                            |');
        putit(15,17,'|____________________________________________|')
      end;                              {declare}


      PROCEDURE TIE;
        begin
          putit(15,11,'|                                           |');
          putit(15,12,'|                                           |');
          putit(15,13,'|                  ...A TIE IS DECLARED     |')
        end;                          {tie}

      PROCEDURE WHO;
        begin
gotoxy(15,11);writeln('|                ',best,');putit(65,11,'|');
          putit(15,12,'|                                           |');
          putit(15,13,'|                  ...IS DECLARED WINNER    |')
        end;                          {who}

      begin
        r1 := roa - bdb; if r1<0 then r1:=0;
        r2 := rob - bda; if r2<0 then r2:=0;
        red := r1 + r2;
        b1 := boa - rdb; if b1<0 then b1:=0;
        b2 := bob - rda; if b2<0 then b2:=0;
        blue := b1 + b2;
        best := 'none';
        if red<blue then best:=players[2];
        if red>blue then best:=players[1];
        declare;
        if best='none' then tie
          else who;
        putit(27,20,'Press RETURN to Continue');
        read(d)
      end;                          {winner}

    PROCEDURE MAYBEWAR;
      var
        name : string; q,smaller,larger : integer;test,done,ans:char;
        r1,sm,la,diff : integer[12];
```

```
PROCEDURE LOGO;
  begin
    clearscreen;
    putit(13,4,'*                                                          *');
    putit(13,5,'*    WW            WW      A          RRRRRR              *');
    putit(13,6,'*     WW    W     WW      AAA         RR   RRR            *');
    putit(13,7,'*      WW   W   WW       AA AA        RRRRRR              *');
    putit(13,8,'*       WW W W WW       AA A AA       RR RR               *');
    putit(13,9,'*        WW   WW       AA     AA      RR   RR             *');
    putit(13,10,'*         W   W       AA       AA    RR     RR           *');
    putit(13,11,'*                                                         *');
    putit(13,12,'*                                                         *')
  end;                              {logo}

PROCEDURE WAR;
  var
    d : char; m : integer;
  begin
    logo;
    m := year + 1;
    if smaller=red then redbudget[m]:=redbudget[m]-1000
      else blubudget[m]:=blubudget[m]-1000;
    putit(13,13,'*     ...has been declared as a result of the          *');
    putit(13,14,'*    disparity between player operational              *');
    putit(13,15,'*    force strengths.                                  *');
    putit(13,16,'*                                                      *');
    gotoxy(13,17);
    writeln('*       As a result of the war ',name);putit(65,17,'*');
    putit(13,18,'*    has had next year''s budget slashed by            *');
    putit(13,19,'*    $1000.                                            *');
    putit(26,21,'<Press RETURN to Continue>');
    read(d)
  end;                                {war}

PROCEDURE UMPLOGO;
  begin
    clearscreen;
    putit(30,5,'**********************');
    putit(30,6,'***    UMPIRE     ***');
    putit(30,7,'**********************');
    putit(18,15,'Do you wish to terminate the game (Y or N)?');
    gotoxy(64,15);
    read(done)
  end;                            {umplogo}

PROCEDURE WARNOTE;
  var
    iii : integer; jjj : char;
  begin
    putit(15,15,'
');
    iii := 100 - trunc(diff);
    gotoxy(17,15);
    writeln('The probability of war is estimated to be ',iii,'%.');
    putit(22,17,'Do you wish a war to ensue (Y or N)?');
    gotoxy(60,17);
    read(jjj);
    if (jjj='y') or (jjj='Y') then
      diff:=99
    else diff:=100
```

```
  end;                              {warnote}

PROCEDURE PUTALL;

  PROCEDURE PUTONE;
    var
      temp1file : file of temp;
    begin
      rewrite(temp1file,'TEMP0:temp01.data');
      with temp1file^ do
        begin
          tyear:=year;tredleft:=redleft;tblueleft:=blueleft;
          traespent:=raespent;tbaespent:=baespent;
          trispent:=rispent;tbispent:=bispent;
          trrdspent:=rrdspent;tbrdspent:=brdspent;
          tnumofsystems:=numofsystems;
          troa:=roa;trob:=rob;trda:=rda;trdb:=rdb;
          tboa:=boa;tbob:=bob;tbda:=bda;tbdb:=bdb;
          tro:=ro;trd:=rd;trrdo:=rrdo;trrdd:=rrdd;
          tfro:=fro;tfrd:=frd;tfrrdo:=frrdo;tfrrdd:=frrdd;
          tbo:=bo;tbd:=bd;tbrdo:=brdo;tbrdd:=brdd;
          tfbo:=fbo;tfbd:=fbd;tfbrdo:=fbrdo;tfbrdd:=fbrdd;
          twarcont:=warcont;tmtr:=mtr;
          tplayers:=players;tredbudget:=redbudget;
          tblubudget:=blubudget
        end;
      put(temp1file);
      close(temp1file,lock)
    end;                            {putone}

  PROCEDURE PUTTWO;
    var
      temp2file : file of systems;
    begin
      rewrite(temp2file,'TEMP0:temp02.data');
      temp2file^ := rs;
      put(temp2file);
      temp2file^ := bs;
      put(temp2file);
      close(temp2file,lock)
    end;                            {puttwo}


begin
  putone;
  puttwo
end;                                {putall}


begin
  r := random mod 101;
  name := players[1];
  smaller := red;
  larger := blue;
  if blue<red then
    begin
      name := players[2];
      smaller  = blue;
      larger := red
    end;
  sm:=smaller;la:=larger;
```

```
    diff := (100*sm) DIV la;
    el:=e;
    umpiego;
    if (done='y') or (done='Y') then
      begin
        clearscreen;
        gotoxy(19,10);
        writeln('Do you wish to resume play from this point');
        gotoxy(19,11);
        write('          at a later time (Y or N): ');
        read(mn);
        if (mn='Y') or (mn='y') then mutall;
        mainflag:=true;
        exit(yearend)
      end
    else
      mainflag:=false;
    if (warcont='y') or (warcont='Y') then
      warnote;
    if el>diff then war
  end;                                  {maybewar}

begin
  winner;
  maybewar
end;                          {yearenn}
```

```
(THE FOLLOWING SEGMENTED PROCEDURES ARE COMPILED AND INCLUDED IN THE MAIN
 PROGRAM AS BATCH 2 USING THE COMPILER 'INCLUDE' OPTION, DUE TO FILE
 SIZE CONSTRAINTS.)


SEGMENT PROCEDURE UMPIRE;
  (Allows game umpire to alter, delete or add systems to game play;
   Also permits umpire to alter player budget figures and probability
   of war.)

    PROCEDURE WRITESYS( J : integer );

      PROCEDURE PARTS;

      PROCEDURE PART1;

        PROCEDURE PART1AUX;
          begin
            writeln('         D       Yrs R&D completed at game start....',rs[J].
yrofRaD);
            writeln('         E       First R&D year cost:.............$',rs[J].
yr1RaDcost);
            writeln('         F       Second R&D year cost:............$',rs[J].
yr2RaDcost);
            writeln('         G       Third R&D year cost:.............$',rs[J].
yr3RaDcost);
            writeln('         H    Earliest year available (after R and D):.....
...',rs[J].yravail)
          end;                        (part1aux)

        begin
          writeln('                                                SYSTEM ',J);
          writeln('         A    Name:.....',rs[J].name);writeln;
          if (rs[J].sort = 'OA') or (rs[J].sort = 'oa') then
            writeln('         B    Type:.....Offensive System Weapon Type A');
          if (rs[J].sort = 'OB') or (rs[J].sort = 'ob') then
            writeln('         B    Type:.....Offensive System Weapon Type B');
          if (rs[J].sort = 'DA') or (rs[J].sort = 'da') then
            writeln('         B    Type:.....Defensive System Weapon Type A');
          if (rs[J].sort = 'DB') or (rs[J].sort = 'db') then
            writeln('         B    Type:.....Defensive System Weapon Type B');
          writeln('         C    First year R&D can start:.....
.',rs[J].yrRaDstarts);
          part1aux
        end;                          (part1)

      PROCEDURE PART2;
        begin
          writeln('         I    Units in inventory (at game start):.....
.',rs[J].inventory);
          writeln('         J    Acquisition cost (per unit):.....
$',rs[J].AQcost);
          writeln('         K    Operating cost (per unit):.....
$',rs[J].OPcost);
          writeln('         L    Value in utils (per unit):.....
.',rs[J].utils);
          writeln('         M    Maximum  annual purchase rate:.....
.',rs[J].purlimit);writeln
        end;                          (part2)

      begin
```

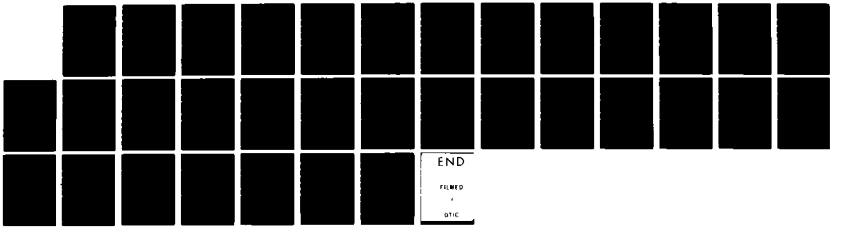
```
      part1;part2
    end;                        {parts}

    begin
      clearscreen;
      parts
    end;                        {writesys}

  PROCEDURE PRINTSYS;
    var
      r : char;f : integer;

    PROCEDURE PART1;

      PROCEDURE PART1AUX;
        begin
          writeln(z,'             Yrs R&D completed at same start....',rs[f].y
rofRaD);
          writeln(z,'             First R&D year cost:.............$',rs[f].y
r1RaDcost);
          writeln(z,'             Second R&D year cost:............$',rs[f].y
r2RaDcost);
          writeln(z,'             Third R&D year cost:.............$',rs[f].y
r3RaDcost);
          writeln(z,'          Earliest year available (after R and D):.....
..',rs[f].yravail)
          end;                      {part1aux}

      begin
        writeln(z,'                                        SYSTEM ',f);
        writeln(z,'             Name:.....',rs[f].name);
        if (rs[f].sort = 'QQ') or (rs[f].sort = 'qa') then
          writeln(z,'             Type:.....Offensive System Weapon Type A');
        if (rs[f].sort = 'QB') or (rs[f].sort = 'qb') then
          writeln(z,'             Type:.....Offensive System Weapon Type B');
        if (rs[f].sort = 'DQ') or (rs[f].sort = 'da') then
          writeln(z,'             Type:.....Defensive System Weapon Type A');
        if (rs[f].sort = 'DB') or (rs[f].sort = 'db') then
          writeln(z,'             Type:.....Defensive System Weapon Type B');
        writeln(z,'          First year R&D can start:.....
',rs[f].yrRaDstarts);
        part1aux
       end;                     {part1}

      PROCEDURE PART2;
        begin
          writeln(z,'          Units in inventory (at same start):.....
',rs[f].inventory);
          writeln(z,'          Acquisition cost (per unit):.....$
',rs[f].AQcost);
          writeln(z,'          Operating cost (per unit):.....$
',rs[f].OPcost);
          writeln(z,'          Value in utils (per unit):.....
',rs[f].utils);
          writeln(z,'          Maximum  annual purchase rate:.....
',rs[f].purlimit);writeln;
          if rs[f].delete = 99 then
            begin
              writeln(z,' ');
              writeln(z,'          ----------------------------------------
```

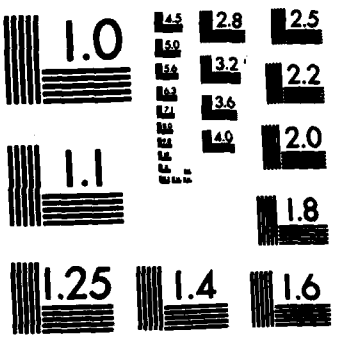
```
***');
                writeln(z,'                *** SYSTEM HAS BEEN DELETED FROM THE GAME
***');
                writeln(z,'                ****************************************
***')
              end;
              writeln(z,' ');writeln(z,' ');writeln(z,' ')
            end;                        {part2}

          begin
            clearscreen;
            putit(15,10,' If you desire a hard copy of all systems to be used');
            putit(15,11,' in the same play, enter ''P'' for printout, otherwise');
            putit(15,12,' press RETURN to exit back to menu.  Be advised that');
            putit(15,13,'   printout will take 5 to 10 mins. and may not be');
            putit(15,14,'                              interrupted.');
            read(r);
            if (r = 'p') or (r = 'P') then
              for f := 1 to numofsystems do
                begin
                  part1;part2
                end
          end;                          {printsys}

        PROCEDURE PRTCHOICE;
          begin
            clearscreen;
            putit(28,1,'*************************');
            putit(28,2,'*** UMPIRE INSTRUCTIONS ***');
            putit(28,3,'*************************');
            putit(31,7,'* Printer Status *');
            putit(10,10,'Enter the number corresponding to current printer status :'
);
            putit(14,13,'1...RS232 UNIT attached and configured to receive data');
            putit(14,15,'2...APPLE SILENTYPE UNIT attached');
            putit(14,17,'3...Printer either not connected or not of option type');
            gotoxy(70,10);
            read(prtr);
            case prtr of
              '1' : rewrite (z,'.RS232');
              '2' : rewrite (z,'.SILENTYPE')
            end
          end;                          {prtchoice}

        PROCEDURE BROWSE;
          var
            i : integer;choice : char;checkflag : boolean;

          PROCEDURE DELETE;
            var
              j : integer;quitflag : boolean;d : char;
            begin
              for j := 1 to numofsystems do
                begin
                  quitflag := false;
                  repeat
                    writesys(j);
                    putit(5,17,'Enter one of the FOLLOWING options:');
                    putit(5,19,'D...Delete displayed system and advance page');
```

```
                putit(5,20,'B...Advance page without deleteing displayed  system
');
                putit(5,21,'C...Quit deleteing');
                read(d);
                case d of
                  'A','a' : begin
                              rs[j].delete := 99;
                              rs[j].wravail := 999;
                              quitflag := true
                            end;
                  'B','b' : quitflag := true;
                  'C','c' : exit(delete);
                  otherwise help
                end
              until quitflag
            end
        end;                                {delete}

      PROCEDURE CHECKRAD( j : integer);
        var
          k, should : integer; ch : char;
        begin
          k := 0;
          if rs[j].wr1RaDcost > 0 then k := 1;
          if rs[j].wr2RaDcost > 0 then k := 2;
          if rs[j].wr3RaDcost > 0 then k := 3;
          if rs[j].wrRaDstarts > 0 then
            begin
              should := rs[j].wrRaDstarts + k;
              if   not (should = rs[j].wravail) and (rs[j].delete = 0) then
                begin
                  putit(1,17,'
');
                  putit(1,18,'
');
                  putit(1,19,'
');
                  putit(1,20,'
');
                  putit(1,21,'
');
                  gotoxy(5,17);
                  writeln('There is an inconsistancy in your R&D parameters. You
r earliest');
                  gotoxy(5,19);
                  writeln('wr available does not allow for ',k,' wrs of R&D besi
nning in wr ',rs[j].wrRaDstarts,'.');
                  putit(17,21,'Press RETURN to correct parameters.');
                  read(c);
                  checkflag := false
                end
            end
        end;                            {checkRaD}

      PROCEDURE ALTER;
        var
          j : integer;d : char;quitflag : boolean;


        PROCEDURE PARAMETER;
          var
```

AD-A124 656 TEMPOR: AN INTERACTIVE SIMULATION FOR THE APPLE III MICROCOMPUTER(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA C D OWENS OCT 82 2/2

UNCLASSIFIED .F/G 9/2 NL





```
    which : char;num : integer;name : string;

  PROCEDURE MAYBE3;
    begin
      case which of
        'J','j' : begin
                    gotoxy(5,18);
                    write('Enter acquisition cost per unit: $');
                    readln(num);
                    rs[j].acost := num
                    end;
        'K','k' : begin
                    gotoxy(5,18);
                    write('Enter operating cost per unit: $');
                    readln(num);
                    rs[j].opcost := num;
                    rs[j].opcostttl := rs[j].opcost*rs[j].inventory
                  end;
        'L','l' : begin
                    gotoxy(5,18);
                    write('Enter util value per unit: ');
                    readln(num);
                    rs[j].utils := num;
                    rs[j].utilsttl := rs[j].utils*rs[j].inventory
                  end;
        'M','m' : begin
                    gotoxy(5,18);
                    write('Enter maximum annual purchase rate: ');
                    readln(num);
                    rs[j].purlimit := num
                  end;
        otherwise help
      end
    end;                    {maybe3}

  PROCEDURE MAYBE2;
    begin
      case which of
        'F','f' : begin
                    gotoxy(5,18);
                    write('Enter 2nd R&D year cost: $');
                    readln(num);
                    rs[j].yr2RaDcost := num
                  end;
        'G','g' : begin
                    gotoxy(5,18);
                    write('Enter 3rd R&D year cost: $');
                    readln(num);
                    rs[j].yr3RaDcost := num
                  end;
        'H','h' : begin
                    gotoxy(5,18);
                    write('Enter year system is first available for oper
ation: ');
                    readln(num);
                    if num = 1 then rs[j].status := 0
                      else rs[j].status := 1;
                    rs[j].yravail := num
                  end;
        'I','i' : begin
```

```
                    gotoxy(5,18);
                    write('Enter units in inventory at start: ');
                    readln(num);
                    rs[j].inventory := num;
                    rs[j].opcost:=rs[j].opcost*rs[j].inventory;
                    rs[j].utilstot := rs[j].utils*rs[j].inventory
                  end;
      otherwise maybe3
    end
  end;                    {maybe2}

PROCEDURE MAYBE1;

  PROCEDURE MAYBE4;
    var
      pick : char;
    begin
      gotoxy(5,18);
      writeln('Select system type: ');
      writeln('      1.   OA...Offensive Weapon Type A');
      writeln('      2.   DA...Defensive Weapon Type A');
      writeln('      3.   OB...Offensive Weapon Type B');
      writeln('      4. . DB...Defensive Weapon Type B');
      read(pick);
      putit(5,18,'                                                    ')
;
      putit(5,19,'                                                    ')
;
      putit(5,20,'                                                    ')
;
      putit(5,21,'                                                    ')
;
      putit(5,22,'                                                    ')
;
      case pick of
        '1' : rs[j].sort := 'OA';
        '2' : rs[j].sort := 'DA';
        '3' : rs[j].sort := 'OA';
        '4' : rs[j].sort := 'DB';
        otherwise help
      end
    end;                    {maybe4}


  begin
    case which of
      'A','a' : begin
                  gotoxy(5,18);
                  write('Enter new system name: ');
                  readln(name);
                  rs[j].name := name
                end;
      'B','b' : maybe4;
      'C','c' : begin
                  gotoxy(5,18);
                  write('Enter year R&D is to start (if necessary): ')
;
                  readln(num);
                  rs[j].yrRaDstarts := num
```

```
                    end;
               'D','d' : begin
                           gotoxy(5,18);
                           write('Enter years R&D completed at same start (norm
ally 0): ');
                           readln(num);
                           rs[j].yrofR&D := num
                         end;
               'E','e' : begin
                           gotoxy(5,18);
                           write('Enter 1st R&D year cost: $');
                           readln(num);
                           rs[j].yr1R&Dcost := num
                         end;
             otherwise maybe2
           end
         end;                    {maybe1}

       begin
         putit(1,17,'
');
         putit(1,18,'
');
         putit(1,19,'
');
         putit(1,20,'
');
         putit(1,21,'
');
         gotoxy(5,17);write('Select a parameter: ');
         read(which);
         maybe1
       end;                      {parameter}

     begin
       for j := 1 to 17 do
         begin
           checkflag := false;
           repeat
             writesys(j);
             if rs[j].delete = 99 then
               begin
                 putit(13,17,'*****************************************');
                 putit(13,18,'*** SYSTEM HAS BEEN DELETED FROM THE GAME ***');
                 putit(13,19,'*****************************************');
                 checkflag := true;
                 putit(13,21,'          Press RETURN to advance page      ');
                 read(d)
               end
             else
               begin
                 putit(5,17,'Enter one of the FOLLOWING options:');
                 putit(10,18,'A...Alter a parameter');
                 putit(10,20,'B...Advance page to next default system');
                 putit(10,21,'C...Quit browsing');
                 read(d);
                 case d of
```

```
                        'A','a' : parameter;
                        'B','b' : begin
                                    checkflag := true;
                                    checkRaD(j)
                                  end;
                        'C','c' : begin
                                    checkflag := true;
                                    checkRaD(j);
                                    if checkflag = true then exit(alter)
                                  end;
                        otherwise help
                      end
                    end
                until checkflag
              end
          end;                              {alter}

        PROCEDURE ADD;
          var
            j,k,num : integer;name : string;id : char;

          PROCEDURE PART1;

            PROCEDURE FILL1;
              begin
                rs[j].status := 0;
                gotoxy(1,5);
                writeln('            First year R&D can start:.....
....',rs[j].yrRaDstarts);
                rs[j].yrofRaD := num;
                gotoxy(1,6);
                writeln('                Yrs R&D completed at same start:...',rs[j]
.yrofRaD)
              end;                          {fill1}

            PROCEDURE PART1AUX;
              var
                pick : char;
              begin
                putit(5,17,'
');
                  gotoxy(5,17);
                  writeln('SELECT SYSTEM TYPE: ');
                  writeln('     1.   OA...Offensive Weapon Type A');
                  writeln('     2.   DA...Defensive Weapon Type A');
                  writeln('     3.   OB...Offensive Weapon Type B');
                  writeln('     4.   DB...Defensive Weapon Type B');
                  read(pick);
                  putit(5,17,'                                                  ')
;
                  putit(5,18,'                                                  ')
;
                  putit(5,19,'                                                  ')
;
                  putit(5,20,'                                                  ')
;
                  putit(5,21,'                                                  ')
;
                  case pick of
                    '1' : begin
```

```
                        rs[j].sort := '09';
                        gotoxy(1,4);
                        writeln('              Type:.....Offensive Weapon Type A
')
                      end;
                '2' : begin
                        rs[j].sort := '0A';
                        gotoxy(1,4);
                        writeln('              Type:.....Defensive Weapon Type A
')
                      end;
                '3' : begin
                        rs[j].sort := '0B';
                        gotoxy(1,4);
                        writeln('              Type:.....Offensive Weapon Type B
')
                      end;
                '4' : begin
                        rs[j].sort := '0B';
                        gotoxy(1,4);
                        writeln('              Type:.....Defensive Weapon Type B
')
                      end;
              otherwise help
            end
        end;                          {partlaux}

      begin
        gotoxy(1,1);
        writeln('                                   NEW SYSTEM ',k);
        putit(5,17,'
');
        gotoxy(10,17);
        write('Enter new system name: ');
        readln(name);
        rs[j].name := name;
        gotoxy(1,3);
        writeln('              Name:.....',rs[j].name);
        partlaux;
        gotoxy(10,17);
        write('Enter year R&D is to start (if no R&D, enter 0): ');
        readln(num);
        rs[j].yrRaDstarts := num;
        if num = 0 then fill1
          else
            begin
              gotoxy(1,5);
              writeln('              First year R&D can start:.............
.........',rs[j].yrRaDstarts);
              putit(5,17,'
');
              gotoxy(10,17);
              write('Enter yrs R&D completed at same start (normally 0): ');
              readln(num);
              if num = 0 then rs[j].status := 1
                else rs[j].status := 2;
              rs[j].yrsfRaD := num;
              gotoxy(1,6);
              writeln('                  Yrs R&D completed at same start:...',r
```

```
s[j].yrofRaD)
            end
          end;                          {part1}

        PROCEDURE PART2;

          PROCEDURE FILL2;
            begin
              rs[j].yr1RaDcost := num;
              gotoxy(1,7);
              writeln('                First R&D year cost:.............$',rs[j]
.yr1RaDcost);
              rs[j].yr2RaDcost := num;
              gotoxy(1,8);
              writeln('                Second R&D year cost:............$',rs[j]
.yr2RaDcost);
              rs[j].yr3RaDcost := num;
              gotoxy(1,9);
              writeln('                Third R&D year cost:.............$',rs[j]
.yr3RaDcost)
            end;                        {fill2}

          begin
            if rs[j].yrRaDstarts = 0 then fill2
              else
                begin
                  putit(5,17,'
          ');
                  gotoxy(10,17);
                  write('Enter 1st R&D year cost: $');
                  readln(num);
                  rs[j].yr1RaDcost := num;
                  gotoxy(1,7);
                  writeln('                First R&D year cost:.............$',r
s[j].yr1RaDcost);
                  putit(5,17,'
          ');
                  gotoxy(10,17);
                  write('Enter 2nd R&D year cost: $');
                  readln(num);
                  rs[j].yr2RaDcost := num;
                  gotoxy(1,8);
                  writeln('                Second R&D year cost:............$',r
s[j].yr2RaDcost);
                  putit(5,17,'
          ');
                  gotoxy(10,17);
                  write('Enter 3rd R&D year cost: $');
                  readln(num);
                  rs[j].yr3RaDcost := num;
                  gotoxy(1,9);
                  writeln('                Third R&D year cost:.............$',r
s[j].yr3RaDcost)
                end
          end;                          {part2}

        PROCEDURE PART3;
          begin
            putit(5,17,'
        ');
            gotoxy(10,17);
```

```
            write('Enter year system is first available for operation: ');
            readln(num);
            rs[j].yravail := num;
            gotoxy(1,10);
            writeln('              Earliest year available (after R and D):.....
..',rs[j].yravail);
            putit(5,17,'
    ');
            gotoxy(10,17);
            write('Enter units in inventory at start: ');
            readln(num);
            rs[j].inventory := num;
            gotoxy(1,11);
            writeln('              Units in inventory (at same start):.....
..',rs[j].inventory);
            putit(5,17,'
    ');
            gotoxy(10,17);
            write('Enter acquisition cost per unit: $');
            readln(num);
            rs[j].acost := num;
            gotoxy(1,12);
            writeln('              Acquisition cost (per unit):.....
.$',rs[j].AQcost)
          end;                        (part3)

        PROCEDURE PART4;
          begin
            putit(5,17,'
    ');
            gotoxy(10,17);
            write('Enter operating cost per unit: $');
            readln(num);
            rs[j].opcost := num;
            gotoxy(1,13);
            writeln('              Operating cost (per unit):.....
.$',rs[j].OPcost);
            putit(5,17,'
    ');
            gotoxy(10,17);
            write('Enter util value per unit: ');
            readln(num);
            rs[j].utils := num;
            gotoxy(1,14);
            writeln('              Value in utils (per unit):.....
..',rs[j].utils);
            putit(5,17,'
    ');
            gotoxy(10,17);
            write('Enter maximum annual purchase rate: ');
            readln(num);
            rs[j].purlimit := num;
            gotoxy(1,15);
            writeln('              Maximum  annual purchase rate:.....
..',rs[j].purlimit);writeln
          end;                        (part4)

        begin
          for j := 18 to 30 do
            begin
```

```
          checkflag := false;
          repeat
            k := j - 17;
            clearscreen;
            part1;
            part2;
            part3;
            part4;
            checkflag := true;
            checkRaD(j)
          until checkflag;
          numofsystems := numofsystems + 1;rs[j].penalty := 0;
          rs[j].delete := 0;rs[j].opcosttt1 := rs[j].opcost*rs[j].inventory;
          rs[j].utilsttl := rs[j].utils*rs[j].inventory;
          putit(10,17,'  Do you wish to create another system? _
');
          putit(10,18,'                    <Y or N>
');
          read(d);
          if (d = 'N') or (d = 'n') then exit(add)
        end
    end;                              {add}

  PROCEDURE BUDGET;
    var
      ii,j,k : integer; budflag : boolean; s : char;

    PROCEDURE CHANGE;
      begin
        putit(1,16,'        —                                                    ');
        putit(1,17,'                                                             ');
        putit(1,18,'                                                             ');
        gotoxy(19,16);
        write('Enter year for change (Then press RETURN): ');
        readln(ii);
        if (ii>20) or (ii<1) then
          begin
            clearscreen;
            putit(24,10,'YEAR MUST BE IN THE RANGE (1-20)');
            putit(27,11,'(Press RETURN to Continue)');
            read(s);
            exit(change)
          end;
        gotoxy(6,17);
        write('Enter new budget, without decimals (Then press RETURN): ');
        readln(j);
        redbudget[ii] := j
      end;                          {change}

    begin
      budflag := false;
      repeat
        clearscreen;
        putit(31,1,'BUDGET INFORMATION');
        putit(20,3,'YEAR     AMOUNT($)     YEAR     AMOUNT($)');
        for ii := 1 to 10 do
          begin
            j := ii + 10;
            k := ii + 4;
            gotoxy(23,k); write(ii);
```

```
            gotoxy(32,k); write(redbudget[i,1]);
            gotoxy(45,k); write(j);
            gotoxy(54,k); write(redbudget[j]);
          end;
        putit(26,18,'<Enter ''C'' to make a change>');
        putit(30,17,'<Enter ''Q'' to quit>');
        gotoxy(38,18);
        read(s);
        case s of
          'Q','q' : budflag := true;
          'C','c' : change;
          otherwise help
        end
      until budflag;
      blubudget := redbudget
    end;                          {budget}

  PROCEDURE ELECT;
    begin
      clearscreen;
      putit(14,2,'       PROBABILITY OF WAR COMPUTATION OPTION        ');
      putit(14,5,'   The annual possibility of a war is determined    ');
      putit(14,6,'probabilistically by a random draw whose threshold  ');
      putit(14,7,'value is based on the magnitude of disparity between');
      putit(14,8,'players'' total adjusted utility figures.            ');
      putit(14,10,'   The umpire may elect to override the above       ');
      putit(14,11,'computation method and be queried annually as to    ');
      putit(14,12,'whether the war event is to take place.             ');
      putit(14,16,'                 OVERRIDE (Y or N) ?                ');
      gotoxy(48,16);read(warcont)
    end;                          {elect}

  begin
    for i := 1 to 100 do
      begin
        clearscreen;
        putit(30,3,'* Systems Review *');
        putit(7,7,'1...Browse/Alter default weapon system budget parameters'
);
        putit(7,9,'2...Add new systems');
        putit(7,11,'3...Delete a system');
        putit(7,13,'4...Browse/Alter annual budgets for all game years');
        putit(7,15,'5...Elect to control the annual probability of war');
        putit(7,17,'6...Obtain printout of all game systems and parameters')
;
        putit(7,19,'7...Quit Umpire Section and begin game play');
        read(choice);
        case choice of
          '1' : alter;
          '2' : add;
          '3' : delete;
          '4' : budget;
          '5' : elect;
          '6' : printsys;
          '7' : begin
                  bs := rs;
                  exit(umpire)
                end;
          otherwise help
        end
      end
  end;                            {browse}

begin
```

```
    prtchoice;
    browse
  end;                          {umpire}

PROCEDURE OPSTATUS ( person : string );
  forward;                      {opstatus}

PROCEDURE AUSTATUS ( person : string );
  forward;                      {austatus}


SEGMENT PROCEDURE BUY (person : string; list : char);
  {Weapon procurement routine; employs checks to ensure budget balance
   is not exceeded, and that purchase rates are not exceeded.  Also allows
   for scrapping of units, and acquisition expenditure updating.}
  var
    gotit : boolean; i, ii, pick : integer; choice : string;
    answer : char;

  PROCEDURE BUYMENU;
    forward;                    {buymenu}

  PROCEDURE BUYHOWMUCH;
      var
        mi,ba,many, left, deficit : integer;budget : array[1..20] of integer;
        d, e : char;espent, ispent, rdspent, expense : integer;

      PROCEDURE CHECK1;
        begin
          gotoxy(15,10);
          writeln('Purchasing ',many,' unit(s)  of ',sys[ii].name,' will excee
d the annual      ');
          writeln('                                purchase rate for this system.
        ');
          gotoxy(18,13);write('The system annual purchase rate : ');
          writeln(sys[ii].purlimit);gotoxy(18,14);
          write('The number remaining for purchase this year : ');
          writeln(sys[ii].unitbuy);
          gotoxy(18,16);
          writeln('Do you desire to purchase fewer than ',many,' units?');
          putit(18,17,'                  <Y or N>                                ');
          read(d);
          if (d = 'N') or (d = 'n') then
            begin
              clearscreen;
              exit(buymenu)
            end
          else
            begin
              putit(18,18,'Reenter the number of system units desired      ');
              putit(18,17,'              <Follow with RETURN>
    ');
              readln(many);
              if many > sys[ii].unitbuy then
                begin
                  putit(18,19,'      YOU''RE NOT COOPERATING!!!');
                  putit(18,20,'       <Press RETURN to exit.>');
                  read(d);
                  clearscreen;
                  exit(buymenu)
```

```
            end
          end
        end;                              {check1}

{THE FOLLOWING SEGMENTED PROCEDURES ARE COMPILED AND INSERTED IN THE MAIN
 PROGRAM AS BATCH 2A USING THE COMPILER 'INCLUDE' OPTION, DUE TO FILE
 SIZE CONSTRAINTS.}


 PROCEDURE CHECK2(much : integer);
          var
            t,k : integer;

          PROCEDURE WORDS;
            begin
              gotoxy(5,18);
              writeln('If you continue with this purchase and do not scrap any o
ther systems');
              gotoxy(5,19);
              writeln('(to lower your operating costs) you will have an annual d
eficit.  Twice');
              gotoxy(5,20);
              writeln('this amount will then be deducted from next year''s budge
t.');
              gotoxy(15,21);
              writeln('Do you wish to continue with this acquisition?');
              gotoxy(20,22);
              writeln('                <Y or N>');
              read(e);
              if (e = 'N') or(e = 'n') then
                begin
                  clearscreen;
                  exit(buymenu)
                end
              else
                clearscreen
            end;                          {words}

          begin
            clearscreen;
            putit(36,1,'WARNING');
            putit(15,2,'You have not enough funds remaining in the year to make'
);
            putit(15,3,'             the desired acquisition.');
            gotoxy(17,5);
            writeln('Budget.....$',budget[year])
;
            gotoxy(17,7);
            writeln('Amount spent previously for acquisitions....$',acspent);
            gotoxy(17,8);
            writeln('Amount spent for intelligence.............$',ispent);
            gotoxy(17,9);
            writeln('Amount spent for R and D..................$',rdspent);
            t := 0;
            for k := 1 to numofsystems do
              if sys[k].delete = 0 then
              t := t + sys[k].opcost[1];
            gotoxy(17,10);
```

```
    writeln('Amount to operate all current');
    gotoxy(17,11);
    writeln('                    inventories.............$',t);
    gotoxy(17,12);
    writeln('Amount remaining.....................$',left);
    gotoxy(17,13);
    writeln('Acquisition and operating costs                  ');
    gotoxy(17,14);
    writeln('          of system quantity selected.....$',much);
    gotoxy(17,16);
    deficit := left - much;
    writeln('Deficit.....................................$',deficit);
    words
  end;                          {check2}

PROCEDURE BUYIT;
  var
    back,min : integer;del : char;
  begin
    if many<0 then
      begin
        clearscreen;
        putit(24,10,'YOU CAN'T BUY NEGATIVE UNITS!');
        putit(24,11,' (Press RETURN to Continue)');
        read(del);
        exit(buyit)
      end;
    back := (sys[ii].inventory+many)-sys[ii].tval;
    if back<0 then back:=0;
    if many<=back then min := many
      else min := back;
    if person = player+1 then
      begin
        redleft := redleft - many*sys[ii].opcost - min*sys[ii].acost;
        raespent := raespent + min*sys[ii].acost;
        rs[ii].unitbuy := rs[ii].unitbuy + many;
        rs[ii].inventory := rs[ii].inventory + many;
        rs[ii].opcosttt := rs[ii].opcosttt + rs[ii].opcost*many;
        rs[ii].utilstt := rs[ii].utilstt + rs[ii].utils*many;
        sys := rs
      end
     else
      begin
        bluleft := bluleft - many*sys[ii].opcost - min*sys[ii].acost;
        baespent := baespent + min*sys[ii].acost;
        bs[ii].unitbuy := bs[ii].unitbuy + many;
        bs[ii].inventory := bs[ii].inventory + many;
        bs[ii].opcosttt := bs[ii].opcosttt + bs[ii].opcost*many;
        bs[ii].utilstt := bs[ii].utilstt + bs[ii].utils*many;
        sys := bs
      end
  end;                          {buyit}

PROCEDURE SCRAPIT;
  var
    back,min : integer;del : char;
  begin
    if many<0 then
      begin
        clearscreen;
```

```
                putit(25,10,'YOU CAN''T SCRAP NEGATIVE UNITS!');
                putit(25,11,'  <Press RETURN to Continue>');
                read(del);
                exit(scrapit)
              end;
            back := sys[ii].inventory-sys[ii].tval;
            if back<0 then back:=0;
            if many<=back then min := many
              else min := back;
            if person = players[1] then
              begin
                redleft := redleft + many*sys[ii].opcost + min*sys[ii].apcost;
                raespent := raespent - min*sys[ii].apcost;
                rs[ii].unitbuy := rs[ii].unitbuy + many;
                rs[ii].inventory := rs[ii].inventory - many;
                rs[ii].opcostttl := rs[ii].opcostttl - rs[ii].opcost*many;
                rs[ii].utilsttl := rs[ii].utilsttl - rs[ii].utils*many;
                sys := rs
              end
            else
              begin
                bluleft := bluleft + many*sys[ii].opcost + min*sys[ii].apcost;
                baespent := baespent - min*sys[ii].apcost;
                bs[ii].unitbuy := bs[ii].unitbuy + many;
                bs[ii].inventory := bs[ii].inventory - many;
                bs[ii].opcostttl := bs[ii].opcostttl - bs[ii].opcost*many;
                bs[ii].utilsttl := bs[ii].utilsttl - bs[ii].utils*many;
                sys := bs
              end
          end;                          {scrapit}

        begin
          if person = players[1] then
            begin
              budget := redbudget;
              left := redleft;
              aespent := raespent;
              ispent := rispent;
              adspent := rrdspent
            end
          else
            begin
              budget := blubudget;
              left := bluleft;
              aespent := baespent;
              ispent := bispent;
              adspent := brdspent
            end;
          gotoxy(15,10);
          if list='c' then
            begin
              writeln('How many units of system ',sys[ii].name,' do you want to
scrap?');
              readln(many);
              scrapit;
              exit(buyhowmuch)
            end;
          writeln('How many units of system ',sys[ii].name,' do you want to purc
```

```
base?');
          readln(many);
          if many > sys[ii].unitbuy then check1;
            ba := (sys[ii].inventory+many)-sys[ii].tval;
            if ba<0 then ba:=0;
            if many<=ba then mi := many
              else mi := ba;
          expense := sys[ii].avcost*mi + sys[ii].opcost*many;
          if expense > left then check2(expense);
          buyit
        end;                              {buyhowmuch}

    PROCEDURE BUYMENU;

      PROCEDURE NEXT1;

        PROCEDURE NEXT2;
          begin
            GOTOXY(38,13);
            write('$',sys[ii].OPcost);
            GOTOXY(47,13);
            write(sys[ii].utils);
            GOTOXY(55,13);
            write(sys[ii].unitbuy);
            GOTOXY(65,13);
            write('$',sys[ii].OPcostTTL);
            GOTOXY(73,13);
            write(sys[ii].utilsTTL);
            putit(1,14,'
    ')
          end;                            {next2}

        begin
          gotoxy(1,7);
          writeln('
');
          writeln('
');
          writeln('
');
          writeln('
');
          writeln('
');
          writeln('
');
          writeln('
');
          putit(4,8,'UPDATED SYSTEM INFORMATION:');
          putit(8,10,'                                    AO      OP                PURCH
TOTAL     ');
          putit(2,11,'SYSTEM(TYPE)    INVENTORY     COST     COST     UTILS     LIMIT
  OPCOST   UTILS ');
          GOTOXY(2,13);
          writeln(sys[ii].name,'(',sys[ii].sort,')');
          GOTOXY(21,13);
          write(sys[ii].inventory);
          GOTOXY(30,13);
          write('$',sys[ii].AOcost);
          next2
```

```
        end;                                    {next1}

      begin
        clearscreen;
        putit(4,1,'SYSTEM INFORMATION:');
        putit(8,3,'                                  AQ        OP                  PURCH          T
OTAL    ');
        putit(2,4,'SYSTEM(TYPE)   INVENTORY    COST    COST    UTILS    LIMIT
 OPCOST  UTILS  ');
        GOTOXY(2,6);
        writeln(sys[ii].name,'(',sys[ii].sort,'):');
        GOTOXY(21,6);
        write(sys[ii].inventory);
        GOTOXY(30,6);
        write('$',sys[ii].AQcost);
        GOTOXY(38,6);
        write('$',sys[ii].OPcost);
        GOTOXY(47,6);
        write(sys[ii].utils);
        GOTOXY(55,6);
        write(sys[ii].unitbuy);
        GOTOXY(65,6);
        write('$',sys[ii].OPcostTTL);
        GOTOXY(73,6);
        write(sys[ii].utilsTTL);
        buyhowmuch;
        next1
      end;                                  {buymenu}

    PROCEDURE WRITEFIRST;
      begin
        gotoxy(15,15);
        writeln('
              ');
        writeln('
              ');
        writeln('
              ');
        writeln('
              ');
        gotoxy(12,15);
        writeln('Enter the name of the system whose inventory you wish to alter.
');
      end;                          {writefirst}

    PROCEDURE WRITESECOND;
      begin
        gotoxy(15,14);
        writeln('
              ');
        writeln('
              ');
        writeln('
              ');
        writeln('
              ');
        writeln('
              ');
        gotoxy(15,15);
        writeln('      Do you want to make another change? ');
```

```
        gotoxy(15,16);
        writeln('                    <Enter Y or N>                    ');
        read(answer);
        if (answer = 'Y') or (answer = 'y') then
          exit(buy)
        else case list of
               'a','c' : exit(opstatus);
               'b' : exit(austatus)
             end
      end;                          {writesecond}

    begin
      for i := 1 to 100 do
        begin
          if i = 1 then writefirst
            else writesecond;
          putit(0,18,'
                      ');
          putit(17,16,'          < Follow with RETURN >                    ');
          putit(27,17,'<   Type 0 to Quit   >');
          readln(choice);
          if (choice = '0') or (choice = 'e')
            then case list of
                   'a','c' : exit(opstatus);
                   'b' : exit(austatus)
                 end;
          ii := 0;
          gotit := false;
          repeat
            ii := ii + 1;
            if (choice = sys[ii].name) or ( ii > numofsystems) then gotit := tru
e;
          until gotit;
          if choice = sys[ii].name then buymen;
            else
              begin
                help;
                exit(buy)
              end
        end
    end;                            {buy}


  SEGMENT PROCEDURE RDSTATUS (person : string);
    {Called from main menu.  Provides for procurement and display of new
     R&D opportunities, shelved R&D projects, and projects in work.
     Allows player to change the status of any project.  Updates
     shelving penalties and all R&D expenditure figures.}
    var
      quitflag : boolean; d : char; title : string; left, much : integer;
      yespent, ispent, rdspent : integer; budget : array [1..20] of integer;

    PROCEDURE RDHEADER(where : integer);
      begin
        clearscreen;
        putit(28,where,title);
        gotoxy(1,3);
        writeln('          YRS   R&D  TOT YRS  YR 1  YR 2  YR 3  RCQ    OP
PURC');
```

```
        writeln('NAME(TYPE)  COMPLETED   REQ''D   COST   COST   COST   COST   COST
UTILS  RATE');
        writeln
      end;                              {writeheader}

    PROCEDURE LISTIT ( i, counter : integer );
      var
        i, many : integer;
      begin
        i := 5 + counter;
        gotoxy(1,i);write(sys[i].name,'(',sys[i].sort,')');
        gotoxy(16,i);write(sys[i].yrofRaD);
        many := 0;
        if sys[i].yr3RaDcost > 0 then many := 3
          else if sys[i].yr2RaDcost > 0 then many := 2
            else if sys[i].yr1RaDcost > 0 then many := 1;
        gotoxy(26,i);write(many);
        gotoxy(32,i);write('$',sys[i].yr1RaDcost);
        gotoxy(38,i);write('$',sys[i].yr2RaDcost);
        gotoxy(44,i);write('$',sys[i].yr3RaDcost);
        gotoxy(50,i);write('$',sys[i].RDcost);
        gotoxy(56,i);write('$',sys[i].OPcost);
        gotoxy(6? ,i);write(sys[i].utils);
        gotoxy(70,i);write(sys[i].purlimit)
      end;                          {listit}

    PROCEDURE CHECK;
      var
        t,n,deficit : integer; c : char;

      PROCEDURE WORDS;
        begin
          gotoxy(2,18);
          writeln('If you continue with this project and do not scrap or shelve
any other systems');
          gotoxy(5,19);
          writeln('(to lower your operating costs) you will have an annual defic
it. Twice');
          gotoxy(11,20);
          writeln('this amount will then be deducted from next year''s budget.')
;
          gotoxy(15,21);
          writeln('Do you wish to continue with this investment?');
          gotoxy(20,22);
          writeln('              (Y or N)');
          read(c);
          if (c = 'N') or(c = 'n') then
            begin
              clearscreen;
              exit(mainstatus)
            end
        end;                            {words}

      begin
        clearscreen;
        putit(36,1,'WARNING');
        putit(15,2,'You have not enough funds remaining in the year to make');
        putit(15,3,'             the desired investment.');
        gotoxy(17,5);
```

```
  writeln('Budget.....$',budget[year]);
  gotoxy(17,7);
  writeln('Amount spent previously for acquisitions....$',acspent);
  gotoxy(17,8);
  writeln('Amount spent for intelligence.....$',ispent);
  gotoxy(17,9);
  writeln('Amount previously spent for R and D.....$',rdspent);
  t := 0;
  for n := 1 to numofsystems do
    if sys[n].delete = 0 then
    t := t + sys[n].opcosttt1;
  gotoxy(17,10);
  writeln('Amount to operate all current');
  gotoxy(17,11);
  writeln('                    inventories.....$',t);
  gotoxy(17,12);
  writeln('Amount remaining.....$',left);
  gotoxy(17,13);
  writeln('R and D costs (including any penalty for          ');
  gotoxy(17,14);
  writeln('    shelving) for weapon system selected....$',much);
  gotoxy(17,16);
  deficit := left - much;
  writeln('Deficit.....$',deficit);
  words
end;                          {check}

  PROCEDURE CRD;
var
  i, counter, where : integer;


PROCEDURE CHOOSE1;
  var
    e : string; j : integer; m : char;

  PROCEDURE MONEY1;
    begin
      if person = players[1] then
        begin
          much := rs[j].wplRaDcost;
          if much > redleft then check;
          redspent := redspent + rs[j].wplRaDcost;
          redleft := redleft - rs[j].wplRaDcost
        end
      else
        begin
          much := bs[j].wplRaDcost;
          if bluleft < much then check;
          bluspent := bluspent + bs[j].wplRaDcost;
          bluleft := bluleft - bs[j].wplRaDcost
        end
    end;                      {money1}

  begin
    putit(19,14,'* NOTE: ALL NEW R&D PROJECTS WILL BE *');
    putit(19,15,'*    SHELVED AT YEAREND UNLESS YOU   *');
    putit(19,16,'*          NOW DESIGNATE             *');
    gotoxy(17,18);
```

```
          writeln('You have $',left,' remaining in this year''s budget');
          putit(7,20,'Enter the name of the system whose R&D you wish to begin:
');
          putit(7,21,'                          <Type ''quit'' to quit>        ');
          gotoxy(65,20);readln(e);
          if (e = 'Quit') or (e = 'quit') then exit(cmd)
          else
            begin
              for i := 1 to numofsystems do
                begin
                  if sys[i].name = e then
                    begin
                      money1;
                      sys[i].status := 2;
                      if person = players[1] then rs := sys
                        else bs := sys;
                      clearscreen;
                      gotoxy(10,10);
                      writeln('R&D funding has been allotted to pursue system ',
e,' development.');
                      putit(10,11,'              <Press RETURN to continue>');
                      read(m);
                      exit(cmd)
                    end
                end;
            end;
          help
        end;                          {choose1}


      begin
        counter := 0;
        where := 1;
        clearscreen;
        title := 'NEW R&D OPPORTUNITIES';
        rdheader(where);
        for i := 1 to numofsystems do
          begin
            if (sys[i].wrRaDstarts <= year) and (sys[i].status = 1) then
              if sys[i].delete = 0 then
                begin
                  counter := counter + 1;
                  listit(i, counter)
                end
          end;
        choose1
      end;                                {cmd}

    PROCEDURE CNRD;
      var
        where, i, counter : integer;

      PROCEDURE CHOOSE2;
        var
          e : string; j : integer; m : char;

        PROCEDURE MONEY2;

          PROCEDURE MONEY2A;
            begin
```

```
        if bs[j].wrofRaD = 0 then
          begin
            brdspent := brdspent - (bs[j].yr1RaDcost + bs[j].penalty);
            bluleft := bluleft + (bs[j].yr1RaDcost + bs[j].penalty)
          end;
        if bs[j].wrofRaD = 1 then
          begin
            brdspent := brdspent - (bs[j].yr2RaDcost + bs[j].penalty);
            bluleft := bluleft + (bs[j].yr2RaDcost + bs[j].penalty)
          end;
        if bs[j].wrofRaD = 2 then
          begin
            brdspent := brdspent - (bs[j].yr3RaDcost + bs[j].penalty);
            bluleft := bluleft + (bs[j].yr3RaDcost + bs[j].penalty)
          end
      end;                          {money2a}

    begin
      if person = players[1] then
        begin
          if rs[j].wrofRaD = 0 then
            begin
              rrdspent := rrdspent - (rs[j].yr1RaDcost + rs[j].penalty);
              redleft := redleft + (rs[j].yr1RaDcost + rs[j].penalty)
            end;
          if rs[j].wrofRaD = 1 then
            begin
              rrdspent := rrdspent - (rs[j].yr2RaDcost + rs[j].penalty);
              redleft := redleft + (rs[j].yr2RaDcost + rs[j].penalty)
            end;
          if rs[j].wrofRaD = 2 then
            begin
              rrdspent := rrdspent - (rs[j].yr3RaDcost + rs[j].penalty);
              redleft := redleft + (rs[j].yr3RaDcost + rs[j].penalty)
            end
        end
      else
        money2a
    end;                        {money2}

  begin
    gotoxy(17,19);
    writeln('You have $',left,' remaining in this year''s budget');
    putit(7,21,'Enter the name of any system whose R&D you wish to shelve:
');
    putit(7,22,'                    (Type ''quit'' to quit)         ');
    gotoxy(66,21);readln(e);
    if (e = 'Quit') or (e = 'quit') then exit(cmnd)
    else
      begin
        for i := 1 to numofsystems do
          begin
            if sys[i].name = e then
              begin
                money2;
                sys[i].status := 3;
                if person = players[1] then rs := sys
                  else bs := sys;
                clearscreen;
```

```
                    gotoxy(10,10);
                    writeln('R&D funding has cut from further pursuit of syste
m ',e,' development.');
                    putit(10,11,'            <Press RETURN to continue>');
                    read(m);
                    exit(cmnd)
                  end
                end
              end;
            help
          end;                    {choose2}

        begin
          counter := 0;
          where := 1;
          clearscreen;
          title := 'CONTINUED R&D OPPORTUNITIES';
          rdheader(where);
          for i := 1 to numofsystems do
            begin
              if (sys[i].wrRaDstarts <= year) and (sys[i].status = 2) then
                if sys[i].delete = 0 then
                  begin
                    counter := counter + 1;
                    listit(i, counter)
                  end
            end;
          choose2
        end;                              {cmnd}

      PROCEDURE SRD;
        var
          where, i, counter : integer; f : char;

        PROCEDURE CHOOSE3;
          var
            e : string; j : integer; m : char;

          PROCEDURE MONEY3;

            PROCEDURE MONEY3R;
              begin
                if bs[j].wrofRaD = 0 then
                  begin
                    much := bs[j].wr1RaDcost + bs[j].penalty;
                    if much > bluleft then check;
                    brdspent := brdspent + (bs[j].wr1RaDcost + bs[j].penalty);
                    bluleft := bluleft - (bs[j].wr1RaDcost + bs[j].penalty)
                  end;
                if bs[j].wrofRaD = 1 then
                  begin
                    much := bs[j].wr2RaDcost + bs[j].penalty;
                    if much > bluleft then check;
                    brdspent := brdspent + (bs[j].wr2RaDcost + bs[j].penalty);
                    bluleft := bluleft - (bs[j].wr2RaDcost + bs[j].penalty)
                  end;
                if bs[j].wrofRaD = 2 then
                  begin
                    much := bs[j].wr3RaDcost + bs[j].penalty;
```

```
                    if much > bluleft then check;
                    brdspent := brdspent + (bs[j].yr3RaDcost + bs[j].penalty);
                    bluleft := bluleft - (bs[j].yr3RaDcost + bs[j].penalty)
                  end
              end;                          {money3a}

            begin
              if person = players[1] then
                begin
                  if rs[j].yrofRaD = 0 then
                    begin
                      much := rs[j].yr1RaDcost + rs[j].penalty;
                      if much > redleft then check;
                      rdspent := rdspent + (rs[j].yr1RaDcost + rs[j].penalty);
                      redleft := redleft - (rs[j].yr1RaDcost + rs[j].penalty)
                    end;
                  if rs[j].yrofRaD = 1 then
                    begin
                      much := rs[j].yr2RaDcost + rs[j].penalty;
                      if much > redleft then check;
                      rdspent := rdspent + (rs[j].yr2RaDcost + rs[j].penalty);
                      redleft := redleft - (rs[j].yr2RaDcost + rs[j].penalty)
                    end;
                  if rs[j].yrofRaD = 2 then
                    begin
                      much := rs[j].yr3RaDcost + rs[j].penalty;
                      if much > redleft then check;
                      rdspent := rdspent + (rs[j].yr3RaDcost + rs[j].penalty);
                      redleft := redleft - (rs[j].yr3RaDcost + rs[j].penalty)
                    end
                end
              else
                money3a
            end;                        {money3}

          PROCEDURE LIST;
            var
              k, count : integer;
            begin
              clearscreen;
              mvtit(10,1,'PENALTY PRICES FOR RESUMING PREVIOUSLY SHELVED R&D PROJE
CTS');
              mvtit(25,3,'System Type                    Penalty');
              count := 5;
              for k := 1 to numofsystems do
                if (sys[k].status = 3) and (sys[k].delete = 0) then
                  begin
                    gotoxy(28,count);write(sys[k].name,'(',sys[k].sort,')');
                    gotoxy(55,count);write('$',sys[k].penalty);
                    count := count + 1
                  end
            end;                        {list}

          begin
            mvtit(15,21,'Enter A to advance page to view penalty prices ');
            mvtit(15,22,'                (Enter 0 to quit)              ');
            read(f);
            if (f = '0') or (f = 'e') then exit(end);
            list;
```

```
          gotoxy(17,19);
          writeln('You have $',left,' remaining in this year''s budget');
          putit(7,21,'Enter the name of any system whose R&D you wish to resume:
');
          putit(7,22,'                    (Type ''quit'' to quit)        ');
          gotoxy(66,21);readln(e);
          if (e = 'Quit') or (e = 'quit') then exit(srd)
          else
            begin
              for i := 1 to numofsystems do
                begin
                  if sys[i].name = e then
                    begin
                      money3;
                      sys[i].status := 2;
                      if person = players[1] then rs := sys
                        else bs := sys;
                      clearscreen;
                      gotoxy(4,10);
                      writeln('R&D funding has resurrected for further pursuit o
f system ',e,' development.');
                      putit(10,11,'              (Press RETURN to continue)');
                      read(m);
                      exit(srd)
                    end
                end;
            help
          end;                        {choose3}

        begin
          counter := 0;
          where := 1;
          clearscreen;
          title := 'SHELVED R&D OPPORTUNITIES';
          rdheader(where);
          for i := 1 to numofsystems do
            begin
              if (sys[i].yrRdStarts <= year) and (sys[i].status = 3) then
                if sys[i].delete = 0 then
                  begin
                    counter := counter + 1;
                    listit(i, counter)
                  end
            end;
          choose3
        end;                              {srd}

      begin
        quitflag := false;
        repeat
          if person = players[1] then
            begin
              left := redleft;
              budget := redbudget;
              aespent := raespent;
              ispent := rispent;
              rdspent := rrdspent
            end
```

```
    else
      begin
        left := blueleft;
        budset := blubudset;
        aespent := baespent;
        ispent := bispent;
        rdspent := brdspent
      end;
    clearscreen;
    putit(20,3,'RESEARCH AND DEVELOPMENT PROJECT STATUS');
    putit(22,8,'A...NEW OPPORTUNITIES');
    putit(22,10,'B...CONTINUING OPPORTUNITIES');
    putit(22,12,'C...SHELVED OPPORTUNITIES');
    putit(22,14,'Q...QUIT');
    read(d);
    case d of
      'A','a' : CRD;
      'B','b' : CNRD;
      'C','c' : SRD;
      'Q','q' : quitflag := true;
      otherwise help
    end
  until quitflag
end;                                    {rdstatus}
```

```
(THE FOLLOWING SEGMENTED PROCEDURES ARE COMPILED AND INSERTED IN THE MAIN
 PROGRAM AS BATCH 28 USING THE COMPILER 'INCLUDE' OPTION, DUE TO FILE
 SIZE CONSTRAINTS.)


 SEGMENT PROCEDURE GETALL;
    (Initializes all parameters if game is a resumption.)

    PROCEDURE GETONE;
      var
        tempifile : file of temp;
      begin
        reset(tempifile,'TEMP0:tempo1.data');
        with tempifile^ do
          begin
            year:=t.year;redleft:=t.redleft;bluleft:=t.bluleft;
            raespent:=t.raespent;baespent:=t.baespent;
            rispent:=t.rispent;bispent:=t.bispent;
            rrdspent:=t.rrdspent;brdspent:=t.brdspent;
            numofsystems:=t.numofsystems;
            roa:=t.roa;rob:=t.rob;rda:=t.rda;rdb:=t.rdb;
            boa:=t.boa;bob:=t.bob;bda:=t.bda;bdb:=t.bdb;
            ro:=t.ro;rd:=t.rd;rrdo:=t.rrdo;rrdd:=t.rrdd;
            fro:=t.fro;frd:=t.frd;frrdo:=t.frrdo;frrdd:=t.frrdd;
            bo:=t.bo;bd:=t.bd;brdo:=t.brdo;brdd:=t.brdd;
            fbo:=t.fbo;fbd:=t.fbd;fbrdo:=t.fbrdo;fbrdd:=t.fbrdd;
            warcont:=t.warcont;prtr:=t.prtr;
            players:=t.players;redbudset:=t.redbudset;
            blubudset:=t.blubudset
          end;
        close(tempifile,lock);
        case prtr of
          '1' : rewrite(z,'.RS232');
          '2' : rewrite(z,'.SILENTYPE')
        end
      end;                                  (setone)

      PROCEDURE GETTWO;
        var
          temp2file : file of systems;
        begin
          reset(temp2file,'TEMP0:tempo2.data');
          rs := temp2file^;
          get(temp2file);
          bs := temp2file^;
          get(temp2file);
          close(temp2file,lock)
        end;                              (settwo)

    begin
      setone;
      settwo
    end;                              (setall)
```

```
{THE FOLLOWING MANAGEMENT PROCEDURES ARE COMPILED AND INSERTED IN THE MAIN
 PROGRAM AS BATCH 1A USING THE COMPILER 'INCLUDE' OPTION, DUE TO FILE
 SIZE CONSTRAINTS.}


 PROCEDURE INVENTORY (a, y : integer);
    {Called by routines Opstatus and Rustatus.  Provides listing of
     system parameters.}
    begin
      y := y + 7;
      GOTOXY(2,y);
      writeln(sys[a].name,'(',sys[a].sort,')',':');
      GOTOXY(21,y);
      write(sys[a].inventory);
      GOTOXY(30,y);
      write('$',sys[a].AOcost);
      GOTOXY(38,y);
      write('$',sys[a].OPcost);
      GOTOXY(47,y);
      write(sys[a].utils);
      GOTOXY(55,y);
      write(sys[a].unitbuy);
      GOTOXY(65,y);
      write('$',sys[a].OPcostTTL);
      GOTOXY(73,y);
      write(sys[a].utilsTTL);
      end;                          {inventory}


  PROCEDURE UPDATE (person : string; amount : integer);
    {Performs annual handshake with players, in turn, and verifies codewords
     for access.}
    var
      d : char;flag : boolean;word : string;
    begin
      clearscreen;writeln;writeln;
      flag := false;
      writeln('                          You are about to begin year ',year,' of TEM
PO.');writeln;
      writeln;
      writeln('                        ',person,' is momentarily to make his/her annu
al ');
      writeln('                                budgetary decisions.');
      writeln;writeln;
        repeat
          write('                        ENTER CODEWORD(cursor remains fixed): ');
          readln(keyboard,word);
          if (person=players[1]) and (word=players[3]) then flag := true
            else if (person=players[2]) and (word=players[4]) then flag := true
              else
                begin
                  gotoxy(0,8,'                                        CODEWORD WRONG! TRY AGAIN
                    ');
                  gotoxy(0,9,'
                ');
                  gotoxy(0,9)
                end;
        until flag;
      writeln;writeln;
      writeln('                        The budget for this year is $',amount);
```

```
      writeln('            (This figure does not reflect inventory operating cos
ts')
      writeln('          for the current year.  See the Main Menu for more infor
mation)');
      writeln;writeln;writeln;writeln;
      writeln('                              Press RETURN to obtain the Main Menu.');
      read(d)
  end;                                (update)

  PROCEDURE WHOPLAYS;
    {Obtains player information from a datafile created by parent chained
     program.}
    var i : integer;
        datafile : file of string;
    begin
      reset(datafile,'TURNKEY1:name.data');
      for i := 1 to 4 do
        begin
          players[i] := datafile^;
          get(datafile);
          {writeln(players[i])}
        end;
      close(datafile)
    end;                          (whoplays)

   PROCEDURE PAGE1(person : string; list : char);
    {Called by Opstatus and Avstatus routines.  Provides for the
     appropriate page headers.}
    var
      choice : char;left : integer;
    begin
      if person = players[1] then
        left := redleft
      else
        left := blueleft;
      gotoxy(15,15);
      writeln('You currently have $',left,' to spend for the fiscal year');
      if list='a' then
          putit(13,18,'(Press P to purchase, S to scrap units or RETURN to exit)
')
        else
          putit(20,18,'(Press P to purchase or RETURN to exit)');
      read(choice);
      case choice of
        'P','p' : buy(person,list);
        'S','s' : begin
                    list:='c';
                    buy(person,list)
                  end;
        ' '     : case list of
                    'a','c' : exit(opstatus);
                    'b' : exit(avstatus)
                  end
        otherwise
          begin
            help;
            case list of
              'a','c' : exit(opstatus);
              'b' : exit(avstatus)
```

```
          end
        end
    end
  end;                                  {page1}

PROCEDURE PAGE2(person : string; list : char);
  {See routine Page1.}
  var
    choice : char;left : integer;

  PROCEDURE ADVANCE;
    begin
      clearscreen;
      case list of
        'a' : putit(28,2,'FORCES CURRENTLY IN INVENTORY');
        'b' : putit(28,2,'FORCES AVAILABLE BUT NOT IN INVENTORY')
      end;
      putit(8,4,'                              AD       OP              PURCH         T
OTAL    ');
      putit(2,5,'SYSTEM(TYPE)   INVENTORY    COST     COST     UTILS    LIMIT
 OPCOST  UTILS  ');
      writeln
    end;                                {advance}

  begin
    if person = players[1] then
      left := redleft
    else
      left := blueleft;
    gotoxy(15,15);
    writeln('You currently have $',left,' to spend for the fiscal year');
    if list='a' then
       putit(7,16,'< Press P to purchase, S to scrap units or press A for nex
t page >')
      else
        putit(16,16,'< Press P to purchase or press A for next page >');
    putit(27,18,'** ADDITIONAL PAGES FOLLOW **');
    read(choice);
    case choice of
      'P','p' : buy(person,list);
      'S','s' : begin
                  list:='c';
                  buy(person,list)
                end;
      'A','a' : advance
      otherwise
        begin
          help;
          case list of
            'a','c' : exit(opstatus);
            'b' : exit(avstatus)
          end
        end
    end
  end;                                  {page2}

PROCEDURE OPSTATUS;
  {Accessed from main menu.  Allows listing of all systems currently in
   inventory.  Provides for system purchases and scrapping through calls
```

```
    to the buy subroutine via Page1 or Page2.)
  var
    d,list : char;i,k,counter : integer;

  begin
    list := 'a';
    for k := 1 to 1000 do
      begin
        counter := 0;
        clearscreen;
        putit(28,2,'FORCES CURRENTLY IN INUENTORY');
        putit(8,4,'                                    AQ       OP                PURCH
TOTAL     ');
        putit(2,5,'SYSTEM TYPE)    INUENTORY     COST     COST     UTILS     LIMIT
  OPCOST  UTILS   ');
        writeln;
        for i := 1 to numofsystems do
          begin
            if (sys[i].yravail <= year) and (sys[i].inventory>0) then
                  if sys[i].delete = 0 then
                    begin
                      counter := counter + 1;
                      if counter > 4 then
                        begin
                          counter := 0;
                          Page2(person,list)
                        end;
                      inventory(i,counter)
                    end;
          end;
        Page1(person,list)
      end
  end;                          {operstus}

PROCEDURE AUSTATUS;
  {Accessed from main menu.  Allows listing of all systems available but
   not yet in inventory. Provides for system purchases and scrapping through
   calls to the buy subroutine via Page1 or Page2.}
  var
    d,list : char;i,k,counter : integer;

  begin
    list := 'b';
    for k := 1 to 1000 do
      begin
        counter := 0;
        clearscreen;
        putit(28,2,'FORCES AVAILABLE BUT NOT IN INUENTORY');
        putit(8,4,'                                    AQ       OP                PURCH
TOTAL     ');
        putit(2,5,'SYSTEM TYPE)    INUENTORY     COST     COST     UTILS     LIMIT
  OPCOST  UTILS   ');
        writeln;
        for i := 1 to numofsystems do
          begin
            if (sys[i].yravail <> year) and (sys[i].inventory = 0) then
                  if sys[i].delete = 0 then
                    begin
                      counter := counter + 1;
```

```
                        if counter > 4 then
                          begin
                            counter := 0;
                            page2(person,list)
                          end;
                        inventory(i,counter)
                      end;
              end;
            page1(person,list)
          end
      end;                          {austatus}


  PROCEDURE CURRENTSTATUS(person : string);
    {Supplies current player budget status.}
    var
      m,i,j,left,aespent,ispent,rdspent : integer;sysbudget : array [1..20] of i
nteger;
      d : char;

    PROCEDURE COSTS;
      begin
        j := 10;
        for i := 1 to numofsystems do
          begin
            if (sys[i].opcosttl > 0) and (sys[i].yravail <= year)  then
              if sys[i].delete = 0 then
                begin
                  j := j + 1;
                  gotoxy(11,j);write(sys[i].name,'(',sys[i].sort,')');
                  gotoxy(30,j);write(sys[i].inventory);
                  gotoxy(49,j);write('$',sys[i].opcosttl)
                end
          end
      end;                          {costs}

    begin
      if person = players[1] then
        begin
          sysbudget := redbudget;left := redleft;
          aespent := raespent;ispent := rispent;rdspent := rrdspent
        end
      else
        begin
          sysbudget := blubudget;left := bluleft;
          aespent := baespent;ispent := bispent;rdspent := brdspent
        end;
      clearscreen;
      putit(30,1,'BUDGET INFORMATION');writeln;gotoxy(17,3);
      write('Year: ',year);gotoxy(32,3);
      write('Total Annual Allowance: $',sysbudget[year]);
      gotoxy(5,5);
      writeln('Amount spent in acquisitions.....$',aespent);
      gotoxy(5,6);
      writeln('Amount spent on intelligence.....$',ispent);
      gotoxy(5,7);
      writeln('Amount spent on R and D.....$',rdspent);
      gotoxy(5,8);writeln('Operating costs by system (those in inventory):');
```

```
  writeln;
  writeln('            System(Type)     Quantity       Total Operating Costs');
  costs;
  m := year + 1;
  gotoxy(10,20);
  writeln('Expected defense budget for next year: $',redbudget[m]);
  gotoxy(24,21);
  writeln('Total Monies Remaining : $',left);
  putit(21,22,'(Press RETURN to return to main menu)');
  gotoxy(1,22);readd)
end;                          {currentstatus}
```

# BIBLIOGRAPHY

Andrus, A., TEMPO Military Planning Game - Rules and Suggestions for Players, working papers, May 1982.

Apple Computer Company, APPLE III PASCAL: Introduction, Filer, and Editor, 1981.

Apple Computer Company, Program Preparation Tools, 1981.

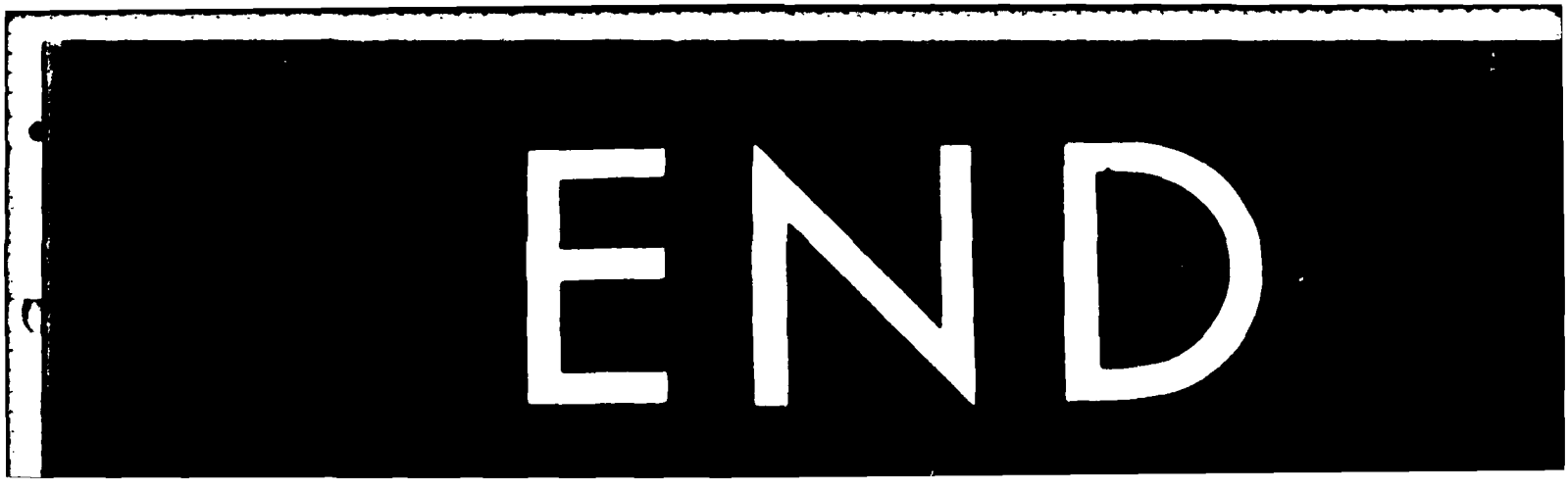
Apple Computer Company, Programmer's Manual, v. 1, 1981.

Apple Computer Company, Programmer's Manual, v. 2, 1981.

Rounce, S. C., An Interactive Military Planning Game for the Naval Postgraduate School Command, Control, and Communications Laboratory, M.S. Thesis, Naval Postgraduate School, Monterey, 1981.

INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Department Chairman, Code 55Zo<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 4. | Professor A. F. Andrus, Code 55As<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 5. | Adjunct Professor G. E. Latta, Code 53Lz<br>Department of Mathematics<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 6. | Associate Professor R. Mendez, Code 53Mu<br>Department of Mathematics<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 7. | Lieutenant Christopher Owens<br>Cruiser Destroyer Group Two<br>FPO New York, New York 09501 | 1 |
| 8. | Office of Naval Research<br>Code 230 Attn: Mr. R. Nagelhout<br>Arlington, Virginia 22217 | 1 |


END